Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Rozpoznávanie cloudových služieb

[Ontosec]

Dokumentácia k riadeniu

Vedúci tímu: Ing. Martin Labaj

Členovia tímu: Denis Grotkovský, Martin Gulis, Marek Vlha, Michal Ševčík, Barbora Ungerová, Jana Tomcsányiová, Jakub Janeček

Školský rok: 2017/2018

Obsah

Úvod

Tento dokument opisuje, ako sme počas semestra pracovali na projekte v rámci predmetu Tímový projekt. Jeho úlohou je simulovať vývoj softvéru v tíme agilným spôsobom (Scrumom). Počas zimného semestra sme pracovali v dvojtýždňových šprintoch. Tímové stretnutia sa konali v pondelky v trvaní 3 hodiny. Ako tím sme sa stretávali aj mimo stretnutí, vždy keď to bolo potrebné, aby sme vyriešili vzniknuté problémy a pomohli si s úlohami.

V tomto dokumente je tiež popísané rozdelenie úloh v tíme a práca jednotlivých členov tímu na častiach projektu.

Dokumentuje aj postup prác na projekte ako aj manažmenty, ktoré boli aplikované pri vývoji.

Predstavenie členov tímu

Jednotlivé roly v tíme (okrem Scrum mastra) sme nepridelili hneď na začiatku semestra. Zvolili sme priebežný postup prideľovania jednotlivých rol, na základe toho, ako sa ktorý člen tímu prejavil. Po prvých dvoch šprintoch mal každý člen tímu pridelenú rolu.

Bc. Jakub Janeček

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Vykonateľné príručky analýzy údajov
- **Záujmy z oblasti informatiky:** analýza údajov, strojové učenie

Rola: Správca webového sídla, Manažér plánovania

Ako scrum master v prvom semestri riadil a dohliadal na vývoj aplikácie, snažil sa riešiť konflikty, ak nejaké boli, aby mohol tím pracovať ako celok. Programuje hlavne funkcionalitu aplikácie, a spracováva všeobecné opisy v dokumentácii.

Bc. Michal Ševčík

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Detekcia žmurkania webkamerou
- **Záujmy z oblasti informatiky:** Počítačová grafika, Game Development

Rola: Scrum master, Správca servera

Spravoval server. Inštaloval potrebné dependencies, programoval backend časť aplikácie. Ovplyvňuje ducha a motiváciu tímu svojou úžasnou náladou, a preto sa v druhom semestri stal scrum masterom.

Bc. Barbora Ungerová

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Grafický editor pre databázu poznatkov

Rola: Správca dokumentácie

Podieľala sa na tvorbe základnej funkcionalite aplikácie. Navrhla a implementovala základný vzhľad aplikácie. Pracovala na implementovaní funkcionality RDF exportu, vytvorenia používateľských rolí a procese publikovania schémy.

Bc. Jana Tomcsányiová

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Školiaci nástroj pre bezpečnostnú technológiu sieťové IDS

Rola: Správca TFS

Napísala metodiku na tvorbu features a user stories a dávala pozor na správne pomenovanie úloh v TFS. Dala dokopy prihlášku a iné dokumenty na TP CUP. Pracovala na výzore dashboard-u certifikačnej schémy a pomáhala s úlohou navrhovania bezpečnostných atribútov a metrík.

Bc. Martin Gulis

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Simulátor formálnych výpočtových strojov

Rola: Správca testovania

Nastavoval automatické testovanie. Napísal metodiku na tvorbu testov. Programoval jednotkové a integračné testy pre značnú časť aplikácie. Implementoval automatické parsovanie viet v aplikácii pomocou natural language processing (NLP).

Bc. Marek Vlha

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Automatizované vyhodnocovanie VHDL modelov

Rola: Správca gitu

Venoval sa správe používateľov, prístupovým právam a pomáhal pri inštalácii servera. Vytvoril metodiku na písanie dokumentácie a pracoval na drobných programátorských úlohách. Navrhol a implementoval výzor domovskej stránky aplikácie a detail porovnania control objective-ov.

Bc. Denis Grotkovský

- Absolvent bakalárskeho štúdia na FIIT STU – Informatika
- **Bakalárska práca:** Publikovanie webových služieb s využitím pre Geografické informačné systémy
- **Záujmy z oblasti informatiky:** Počítačová grafika, dizajn

Rola: Správca kódu

Podieľal sa na vytváraní funkcionality aplikácie. Vytvoril plagát na TP CUP a chybové stránky v aplikácii. Pracoval na lokalizácii, preklade aplikácie a na možnosti schvaľovania novovytvorených entít. Tiež písal pár metodík.

Podiel práce

Tabuľka 1 dokumentuje podiel práce členov tímu na častiach dokumentácie k riadeniu projektu.

Tabuľka 1: Podiel práce v dokumentácii k riadeniu projektu.

Časť dokumentu	Člen(ovia) tímu
Úvod	Bc. Jakub Janeček
Manažment komunikácie	Bc. Jakub Janeček
Manažment vytvárania úloh	Bc. Jana Tomcsányiová
Manažment štýlu kódu - Python	Bc. Jakub Janeček
Manažment testovania	Bc. Martin Gulis
Manažment gitu	Bc. Barbora Ungerová
Manažment dokumentácie	Bc. Marek Vlha
Manažment chýb	Bc. Michal Ševčík
Sumarizácia šprintov	Bc. Jana Tomcsányiová
Retrospektíva Šprint 1	Bc. Martin Gulis
Retrospektíva Šprint 2	Bc. Barbora Ungerová
Retrospektíva Šprint 3	Bc. Jakub Janeček
Retrospektíva Šprint 4	Bc. Denis Grotkovský
Retrospektíva Šprint 5	Bc. Jakub Janeček
Retrospektíva Šprint 6	Bc. Michal Ševčík
Retrospektíva Šprint 7	Bc. Michal Ševčík
Retrospektíva Šprint 8	Bc. Michal Ševčík
Retrospektíva Šprint 9	Bc. Michal Ševčík
Retrospektíva Šprint 10	Bc. Michal Ševčík
Retrospektíva Šprint 11	Bc. Michal Ševčík
Globálna retrospektíva ZS	Bc. Jakub Janeček
Globálna retrospektíva LS	Bc. Jana Tomcsányiová
Metodiky – kontrola	Bc. Jana Tomcsányiová
Metodika na code review	Bc. Denis Grotkovský
Metodika k lokalizácii a prekladu	Bc. Denis Grotkovský
Metodika na štýl kódu	Bc. Jakub Janeček
Metodika na tvorbu features a user stories	Bc. Jana Tomcsányiová
Metodika na tvorbu testov	Bc. Martin Gulis
Metodika na zdieľaný vývoj pomocou gitu	Bc. Barbora Ungerová

Metodika na písanie dokumentácie	Bc. Marek Vlha
Metodika na manažment chýb	Bc. Michal Ševčík
Evidencia úloh	Bc. Jana Tomcsányiová

Tabuľka 2 opisuje podiel práce členov tímu na častiach dokumentácie k inžinierskemu dielu.

Tabuľka 2: Podiel práce v dokumentácii k inžinierskemu dielu.

Časť dokumentu	Člen(ovia) tímu
Úvod	Bc. Jakub Janeček
Globálne ciele	Bc. Jakub Janeček
Celkový pohľad na systém	Bc. Jakub Janeček 80% Bc. Michal Ševčík 20%
Moduly – kontrola	Bc. Jana Tomcsányiová
Správa používateľov	Bc. Jakub Janeček
Vytváranie a správa opisu certifikačnej Schémy ontológiou	Bc. Jakub Janeček 85% Bc. Martin Gulis 15%
ElasticSearch	Bc. Michal Ševčík
RDF export	Bc. Barbora Ungerová
Porovnanie certifikačných schém	Bc. Jakub Janeček Bc. Michal Ševčík
Server	Bc. Michal Ševčík
Správa cloudových služieb	Bc. Jakub Janeček
Automatické vytváranie opisu certifikačnej schémy	Bc. Jakub Janeček Bc. Michal Ševčík Bc. Marek Vlha Bc. Jana Tomcsányiová
NLP	Bc. Martin Gulis
Inštalačná príručka	Bc. Michal Ševčík
Používateľské príručky – screenshoty a kontrola	Bc. Jana Tomcsányiová
Obnovenie hesla	Bc. Barbora Ungerová
Porovnanie certifikačných schém	Bc. Barbora Ungerová
RDF export	Bc. Barbora Ungerová
Schvaľovanie metrík a bezpečnostných atribútov	Bc. Barbora Ungerová
Schvaľovanie schém	Bc. Marek Vlha
Správa používateľov	Bc. Barbora Ungerová
Správa používateľských práv	Bc. Barbora Ungerová
Generovanie control objective-ov	Bc. Michal Ševčík
Vytvorenie nového control objective-u a otázky control-u	Bc. Barbora Ungerová

Vytvorenie nového control-u	Bc. Barbora Ungerová
Vytvorenie novej certifikačnej schémy	Bc. Barbora Ungerová
Importovanie novej certifikačnej schémy	Bc. Michal Ševčík
Správa cloudových služieb	Bc. Jakub Janeček (aj screenshoty)
Technická dokumentácia	Bc. Michal Ševčík Bc. Martin Gulis

Tabuľka 3 obsahuje podiel práce členov tímu na jednotlivých častiach projektu.

Tabuľka 3: Podiel práce na častiach projektu.

Časť projektu	Člen(ovia) tímu
Server a jeho nastavenie	Bc. Michal Ševčík
Monitorovanie výkonu servera	Bc. Michal Ševčík
Sledovanie chýb backend/frontend	Bc. Michal Ševčík
ElasticSearch	Bc. Michal Ševčík
Fulltext vyhľadávanie metrík a bezpečnostných atribútov	Bc. Michal Ševčík
Continuous Integration	Bc. Michal Ševčík 70% Bc. Martin Gulis 30%
Team Web	Bc. Michal Ševčík (1. semester) Bc. Jakub Janeček (2. semester)
Vytvorenie Staging vetvy	Bc. Michal Ševčík
Lokalizácia	Bc. Denis Grotkovský
Testovanie	Bc. Martin Gulis
Dizajn aplikácie	Bc. Barbora Ungerová
Domovská stránka aplikácie	Bc. Marek Vlha
RDF export	Bc. Barbora Ungerová 60% Bc. Martin Gulis 40%
Správa cloudových služieb	Bc. Jakub Janeček
Prihlásenie používateľa	Bc. Jakub Janeček 80% Bc. Martin Gulis 20%
Vytváranie ďalších používateľských kont	Bc. Marek Vlha 90% Bc. Martin Gulis 10%
Obnova hesla používateľa	Bc. Denis Grotkovský 90% Bc. Martin Gulis 10%
Zmena hesla používateľa	Bc. Marek Vlha 90% Bc. Martin Gulis 10%
Registrácia používateľa	Bc. Jakub Janeček
Vytvorenie rolí	Bc. Barbora Ungerová
Prehľad certifikačných schém	Bc. Jakub Janeček 85%

	Bc. Martin Gulis 15%
Vytvorenie novej certifikačnej schémy	Bc. Barbora Ungerová Bc. Jakub Janeček Bc. Martin Gulis
Importovanie certifikačnej schémy	Bc. Michal Ševčík
Detail certifikačnej schémy	Bc. Jakub Janeček 85% Bc. Martin Gulis 15%
Prehľad otázok control-u	Bc. Barbora Ungerová
Výzor dashboard-u certifikačnej schémy	Bc. Jana Tomcsányiová Bc. Martin Gulis
Nastavenie prístupových práv pre schému	Bc. Jakub Janeček Bc. Martin Gulis Bc. Barbora Ungerová
Proces publikovania certifikačnej schémy	Bc. Marek Vlha Bc. Barbora Ungerová
Porovnanie certifikačných schém	Bc. Jakub Janeček Bc. Michal Ševčík
Detail porovnania control objective-ov	Bc. Marek Vlha
Vytvorenie nového control-u	Bc. Denis Grotkovský Bc. Jakub Janeček Bc. Martin Gulis
Detail control-u	Bc. Jakub Janeček 85% Bc. Martin Gulis 15%
Vytvorenie nového control objective-u	Bc. Jakub Janeček 90% Bc. Martin Gulis 10%
Detail control objective-u	Bc. Martin Gulis
Zaznamenávanie nových pridaných metrík a bezpečnostných atribútov	Bc. Jakub Janeček
Posúdenie control-ov a control objective-ov	Bc. Denis Grotkovský
Schvaľovanie metrík a bezpečnostných atribútov	Bc. Denis Grotkovský
NLP – spolupráca na každej z NLP úloh	Bc. Martin Gulis
Automatické vytváranie opisu certifikačnej schémy	Bc. Jakub Janeček
Automatické vytváranie bezpečnostného atribútu a metriky	Bc. Michal Ševčík Bc. Marek Vlha
Navrhovanie bezpečnostných atribútov a metrík	Bc. Jana Tomcsányiová
Generovanie opisu control objective-ov	Bc. Michal Ševčík

Aplikácie manažmentov

Táto časť obsahuje opis manažmentov, ktoré boli aplikované počas vývoja aplikácie, a taktiež opis s nimi súvisiacich používaných nástrojov.

Manažment komunikácie

Komunikácia je pravdepodobne najdôležitejším aspektom práce v tíme. Bez vhodnej formy a pravidiel nie je možné efektívne komunikovať, a teda efektívne pracovať na vývoji. Pri našej komunikácii sme využívali hlavne nástroje TFS a Slack, a samozrejme osobné stretnutia.

TFS (Team Foundation Server)

TFS sme využívali na plánovanie a monitorovanie šprintov. Každý člen v ňom zaznamenával svoju prácu a aktivitu. TFS taktiež poskytoval možnosti integrácie gitu, Slack-u a iných nástrojov, ktoré sme využívali pri vývoji. Komunikácia medzi členmi tímu v tomto nástroji prebiehala do istej miery pasívne, kde členom umožnila lepšie organizovať náväzné úlohy, monitorovať potreby členov, keď nastali komplikácie s úlohami a rôzne iné. Pomocou tohto nástroja sme vykonávali aj code review. Umožňoval taktiež export úloh.

Tím si na tento nástroj zvykol celkom rýchlo a je s ním spokojný. Chvíľu trvalo, kým sme sa naučili, kde je potrebné čo zaznamenávať, aby sme vyťažili čo najviac z ponúkaných možností, ale aktuálne sa nám to už darí.

Slack

Slack sme využívali ako hlavný komunikačný nástroj. Najprv sme si vytvorili niekoľko základných kanálov. Postupne vznikali požiadavky na ďalšie kanály, tak sme ich vytvorili. Aktuálne existujú nasledovné kanály:

- **alarms** – monitorovanie prostriedkov servera (automatické výpisy z *netdata*),
- **general –** základné dianie, bežné rozhodnutia a informácie,
- **issues** – chyby v aplikácii (automatické hlásenia z *Rollbar*),
- **problems –** problémy s úlohami alebo s čímkoľvek iným,
- **process –** zlepšenia metodík a procesu vývoja,
- **reminders –** pripomienky (naháňanie členov Scrum mastrom),
- **server –** nastavovanie servera,
- **tutorials -** odkazy na tutoriály, postupy nastavení a i.,
- **standups** – záznamy so stand-up stretnutí,

- **timeline -** feed z TFS.

Tím bol so Slack-om spokojný, jeho funkcionalita bola absolútne postačujúca na všetko potrebné.

Stretnutia

Tímové stretnutia sa konali v pondelok od 13:00 do 16:00. Na týchto stretnutiach tím plánoval šprinty, riešil problémy v strede šprintov, zaznamenával retrospektívy, a rozprával sa o všetkom možnom, čo súviselo s vývojom v tíme. Tieto stretnutia boli veľmi prospešné a veľa sme sa na nich naučili.

Ako tím sme sa občas stretávali aj mimo stretnutí, aby sme sa navzájom informovali o stave našich úloh a umožnili ich lepšiu koordináciu.

Manažment vytvárania úloh

Pre dodržiavanie správneho názvoslovia a popisu Features, User Stories a úloh bola vytvorená Metodika na tvorbu Features a User Stories.

Už pri plánovaní prvého šprintu a vytváraní User Stories začali byť ich názvy nekonzistentné a zabúdalo sa na opis jednotlivých User Stories. Preto sme usúdili, že je potrebné vytvoriť metodiku, pomocou ktorej zabezpečíme potrebnú konzistenciu. Taktiež sme zistili, že je žiadúce mať pravidlá na to, akým spôsobom sa v rámci jednotlivých úloh vypĺňajú časti Original Estimate, Remaining Work a Completed Work.

Nejaký čas trvalo, kým si členovia tímu zvykli na to, ako správne písať názvy a vypĺňať počet hodín v rámci úloh, ale aktuálne už nemajú problém s dodržiavaním tejto metodiky.

Manažment štýlu kódu – Python

Pre štýl kódu bola vytvorená metodika Metodika na štýl kódu – Python takmer na začiatku práce na projekte. Dôvodom bolo, aby sa zaviedli konvencie na písanie kódu, čo by zlepšilo jeho čitateľnosť medzi členmi, a predišlo sa nezhodám čo ako zapisovať.

Táto pôvodná myšlienka čiastočne zlyhala, keďže sa na začiatku nedodržali niektoré pravidlá. Usúdili sme však, že na začiatku projektu je dôležitejšie, aby si všetci členovia tímu zvykli na Python a Django, keďže pre väčšinu to bola nová skúsenosť. Postupne po niekoľkých týždňoch sme nabaľovali pravidlá z metodiky a dbali na ich dodržiavanie. Taktiež samotná metodika sa rozširovala podľa potreby s pribúdajúcimi novými skúsenosťami.

Dohodli sme sa, že na statickú analýzu budeme používať nástroj Pylint, ktorý sa dá nakonfigurovať, a dokáže odhaliť porušenia pravidiel. Keďže zo začiatku projektu nám ostal kód, ktorého forma nebola vyhovujúca pravidlám z metodiky, naplánovali sme si úlohu refaktorizácia kódu, ktorá okrem iného zahŕňala aj úpravu existujúceho kódu podľa pravidiel z metodiky.

Manažment testovania

Spočiatku bolo dôležité nastaviť continuous integration a prostredie testovania na serveri pre automatické testovanie. Počas toho sme študovali možnosti testovania, ktoré nám ponúka Python a Django. V prvých fázach testy prebiehali iba lokálne a v čase úspešnej konfigurácie na serveri už niektoré testy existovali.

Pre testovanie vznikla Metodika na tvorbu testov. Obsahuje pravidlá a návody k písaniu testov ako aj potrebné programy na spúšťanie testov lokálne pre každého člena tímu. Metodika sa stále rozširuje s prípadnými zmenami v súvislosti s testami.

Najskôr sa začínalo jednoduchými jednotkovými testami, pre ktoré stačilo využívať prostriedky poskytované pre základné Python a Django aplikácie. Postupne sa však začínalo rozvíjať aj integračné testovanie, na ktoré bola využitá automatizácia webového prehliadača prostredníctvom Selenium a ChromeDriver. Pri vývoji testov sa rýchlo prišlo na to, že sú niektoré testy podobné. Preto sa postupne začala vytvárať hlavná kostra pre znovupoužiteľné časti testov, akými bolo správne otvorenie webového prehliadača a podobne. Keďže aplikácia sa neustále vyvíja, je nutné testy postupne dopĺňať.

Manažment gitu

Pre zdieľanie kódu na gite bola vytvorená metodika Metodika na zdieľaný vývoj pomocou gitu. Postupne sa táto metodika vyvíjala. Dôvodom vzniku metodiky bolo, aby sa zaviedli pravidlá pre zdieľanie kódu na gite, pravidlá k jednotlivým popisom kódu ako aj riešenie konfliktov vedľajších vetiev s hlavnou.

Metodiku sa nepodarilo úplne dodržať, kvôli jej postupnému vývoju. Postupom času sa dohodlo na jazyku a forme opisu správ ako aj riešení konfliktov medzi vetvami. Všetci členovia tímu sa však týmto zmenám postupne prispôsobovali a snažili sa pokyny priebežne dodržiavať.

Na začiatku metodika obsahovala základný popis pre zdieľanie kódu na gite vo vetvách, riešení konfliktov a základné opisy. Postupne sa dohodlo na jazyku na písanie správ do gitu ako aj ich formáte a obsahu. V metodike pribudol aj návod na používanie potrebných funkcií.

Manažment dokumentácie

Keďže každé inžinierske dielo si vyžaduje dokumentáciu, bolo nutné vytvoriť isté pravidlá, ktorými sa členovia tímu musia držať pri písaní dokumentácie. Tieto pravidlá sú stručne obsiahnuté v metodike: Metodika na písanie dokumentácie.

Metodika počas vzniku obsahovala rôzne pokyny, ktoré sa menili podľa dohadovania sa tímu. Táto metodika vznikla na základe rôznych dokumentácií tímu. Bolo teda nutné sa dohodnúť na pravidlách písania dokumentácie, formátovania textu, veľkosti písma a podobne.

Metodika obsahuje hlavne základné pravidlá ako veľkosť písma, typ písma, ale aj ktoré veci je nutné dopisovať do jednotlivých dokumentov.

Manažment chýb

V našej webovej aplikácii monitorujeme chyby pomocou systému Rollbar. Bolo teda potrebné oboznámiť tím s tým, ako pracovať s týmto systémom a ako sa chovať k vzniknutej chybe.

Toto obsahuje Metodika na manažment chýb. Keďže je táto metodika ešte pomerne čerstvá, jej efektivitu a dodržiavanie zatiaľ nevieme vyhodnotiť.

Metodika opisuje základný životný cyklus chyby a z časti opisuje prácu so systémom Rollbar.

Sumarizácia šprintov

V tejto časti je stručne popísaná sumarizácia šprintov, ktorými tím prešiel počas zimného semestra. Informácie o stave úloh po jednotlivých šprintoch a členovia tímu zodpovední za splnenie úloh sa nachádzajú v časti Evidencia úloh.

Šprint 1 – Autíčko

Do prvého šprintu, ktorý začal 2. 10. 2017 a skončil 16. 10. 2017, sme si naplánovali nasledujúce úlohy:

1. Prihlásenie administrátora,
2. Vytvorenie novej certifikačnej schémy,
3. Zobrazenie prehľadu certifikačných schém,
4. Zobrazenie certifikačnej schémy,
5. Vytvorenie control-u certifikačnej schémy,
6. Nainštalovanie servera,
7. Vytvorenie webovej stránky tímu,
8. Pripravenie aplikačného servera,
9. Vytvorenie automatického testovania a nasadzovania,
10. Vytvorenie metodiky na tvorbu features a user stories,
11. Vytvorenie metodiky na tvorbu testov,
12. Vytvorenie metodiky na štýl kódu,
13. Vytvorenie metodiky na zdieľaný vývoj pomocou gitu,
14. Vytvorenie metodiky na code review.

Z týchto úloh sme úspešne splnili všetky okrem prvej – Prihlásenie administrátora – ktorá sa preniesla do nasledujúceho šprintu. Cieľom prvého šprintu bolo taktiež rozbehnutia vývojového prostredia lokálne jednotlivými členmi tímu so všetkými jeho súčasťami a oboznámenie sa s vývojom v ňom.

Retrospektíva

Čo sa nám páčilo:

- členovia tímu si pomáhali,
- v časovej tiesni dokázali všetci dať zo seba maximum.

Čo sa nám nepáčilo:

- nadhodnotenie niektorých úloh, podhodnotenie iných,
- problémy s prostredím,
- chyby sa nehlásili a neriešili globálne,

- chaos na gite,
- niektorí členovia tímu začali pracovať na svojich úlohách veľmi neskoro,
- niektorí členovia tímu si nepozreli potrebné tutoriály,
- neriadenie sa metodikami, slabá návodovosť metodík,
- nenasadenie na serveri.

Čo by sme zlepšili:

- ohodnotenie úloh,
- príjemnejšie riešenie problémov,
- aby ľudia, ktorí pracujú na úlohách, od ktorých závisia iné úlohy, spravili svoje úlohy čo najskôr,
- preštudovanie tutoriálov všetkými členmi tímu,
- lokálne rozbehnutie aplikácie všetkými členmi tímu,
- člen tímu by sa chcel zaoberať aj programovaním v Pythone a nie iba nastavovať server.

Šprint 2 – Bábika

V rámci druhého šprintu, v trvaní od 16. 10. 2017 do 30. 10. 2017, sme pracovali na týchto úlohách:

1. Prihlásenie administrátora,
2. Aktualizovanie metodík,
3. Vytvorenie automatického testovania,
4. Vytvorenie automatického nasadzovania,
5. Prihlásenie do TP CUP-u,
6. Vytvorenie metodiky na dokumentáciu,
7. Vytváranie dalších používateľských kont,
8. Preloženie aplikácie do angličtiny,
9. Vytvorenie rozloženia a menu,
10. Vytvorenie a zobrazenie prehľadu control objective-ov.

Všetky tieto úlohy sa nám podarilo úspešne dokončiť a odovzdať na konci šprintu Product Owner-om. Cieľom šprintu bolo hlavne vytvoriť základnú funkcionalitu umožňujúcu opísať certifikačnú schému pomocou ontológie, a taktiež automatizovať proces nasadzovania stránky na server.

Retrospektíva

Čo sa nám páčilo:

- burn-down chart išiel pekne postupne dole,

- spravenie všetkých úloh v šprinte,
- nasadenie stránky na server,
- aktívnejšia práca na projekte,
- množstvo urobenej roboty.

Čo sa nám nepáčilo:

- zjednocovanie html súborov, ktoré si každý najprv písal, ako sa mu páčilo,
- nejasné dohadovanie sa na termíne tímových stretnutí,
- práca s javascriptom,
- robenie rebase projektu,
- nedostatočná dokumentácia.

Čo by sme zlepšili:

- dokumentácia k projektu,
- po ukončení User Story ju treba presunúť do stĺpca Resolved, aby sa Product Owner na ňu mohol pozrieť a zhodnotiť ešte pred koncošprintovým stretnutím,
- rozloženie úloh na členov tímu,
- priebežné vypĺňanie Completed Work.

Šprint 3 – Céčka

Náplňou tretieho šprintu, ktorý prebiehal od 30. 10. 2017 do 13. 11. 2017, boli tieto úlohy:

1. Zmena hesla používateľa,
2. Obnova hesla,
3. Správa prístupu k editácii schémy,
4. Zaznamenávanie novopridaných metrík,
5. Full-text vyhľadávanie existujúcich metrík,
6. Zaznamenávanie novopridaných atribútov,
7. Full-text vyhľadávanie existujúcich atribútov,
8. Dashboard certifikačnej schémy,
9. Refaktorovanie testov,
10. Refaktorovanie dokumentácie.

Všetky úlohy sme zvládli ukončiť a odovzdať. Cieľom šprintu bolo hlavne upraviť funkcionalitu pre opis certifikačných schém o ďalšie možnosti, správu používateľov a taktiež prepracovať automatické testovanie aplikácie.

Retrospektíva

Čo sa nám páčilo:

- prístup tímu,
- prístup Scrum Mastra,
- dokončenie šprintu v slušnom stave,
- Martin G. spravil testy pre aplikáciu,
- množstvo vykonanej práce.

Čo sa nám nepáčilo:

- komunikácia v tíme,
- robiť merge ako posledný člen tímu,
- členovia tímu nemysleli dopredu vzhľadom na časový manažment,
- nekomunikovanie nestíhania spravenia úloh,
- priebeh šprintu,
- „burn-up“ chart,
- iba 1 User Story v resolved na konci šprintu.

Čo by sme zlepšili:

- časový manažment niektorých členov tímu,
- komunikácia medzi členmi tímu,
- byť konzistentný v názvoch súborov,
- plánovanie šprintu zo strany členov tímu,
- dorobiť User Story a začať ďalšiu až potom,
- informácie k User Stories zapisovať do TFS,
- pýtať sa Product Owner-ov na nejasnosti,
- pracovať priebežne,
- nehádať sa na stretnutiach.

Šprint 4 – Dinosaurus

Štvrtý šprint trval od 13. 11. 2017 do 27. 11. 2017 a boli v rámci neho naplánované tieto úlohy:

1. TP Mentoring,
2. Secure server,
3. Zobrazenie porovnanie medzi dvoma schémami,
4. Rola reviewera opisu schémy,
5. Schvaľovanie nových metrík,
6. Schvaľovanie nových security atribútov,
7. Exportovanie zadaných údajov do RDF,
8. Publikovanie opisu schémy,
9. Refaktorovanie kódu aplikácie,

10. Sledovanie chýb v backend-e,
11. Sledovanie výkonu servera.

Splnili sme všetky úlohy okrem poslednej – Sledovanie výkonu servera. Táto úloha sa presunula do ďalšieho šprintu.

Retrospektíva

Čo sa nám páčilo:

- členovi tímu sa páčila úloha, ktorú mal pridelenú,
- členovia tímu si píšu testy pre svoju úlohu,
- burndown chart išiel naozaj smerom dole.

Čo sa nám nepáčilo:

- málo času popri iných projektoch,
- to, ako niektorí členovia tímu robili code review,
- písanie testov,
- členovia tímu medzi sebou nekomunikujú,
- členovia tímu nedávajú spätnú väzbu.

Čo by sme zlepšili:

- nastavenia aplikácie,
- časový manažment členov tímu,
- komunikácia v tíme,
- viac pracovať v prvom týždni,
- dokumentácia,
- dokončenie testov,
- štruktúrovanie kódu,
- komentovanie kódu,
- grafické používateľské rozhranie.

Šprint 5 – Elektrický vláčik

Počas piateho šprintu od 27. 11. 2017 do 11. 12. 2017 sme pracovali na nasledujúcich úlohách:

1. Sledovanie výkonu servera,
2. Sledovanie chýb vo frontende,
3. Dokončenie šprintu 4,
4. Upravenie funkcionality priraďovania práv ku schéme,
5. Upravenie RDF funkcionality,

6. Finalizácia projektu na odovzdanie,
7. Upravenie schvaľovania nových entít,
8. Upravenie zobrazenia porovnania schém,
9. Vytvorenie metodiky na manažment chýb,
10. Aktualizácia jquery,
11. Zmenenie štruktúry kódu,
12. Finalizácia dokumentácie na odovzdanie.

Z týchto úloh sa nám počas šprintu nepodarilo splniť úlohu Zmenenie štruktúry kódu.

Retrospektíva

Čo sa nám páčilo:

- stránka vyzerá celkom dobre,
- členovi tímu sa páčila úloha, na ktorej pracoval,
- funkcionalita aplikácie,
- funkčnosť pridávania control objective a ich porovnanie,
- monitorovací prostriedok servera,
- automatické testovanie,
- diskusia v tíme,
- identifikovanie problémov v retrospektíve.

Čo sa nám nepáčilo:

- nestíhanie robenia code review a úloh,
- nedostatok času,
- umelo vytvorené chyby od vedúceho tímu za účelom otestovania vzniknutej Metodiky manažmentu chýb,
- automatické testy trvajú pridlho,
- pomalý rozbeh šprintu,
- prideľovanie code review,
- nehlásenie vzniknutých problémov,
- hádanie sa členov tímu namiesto identifikovania problémov,
- tím má súkromné stand up-y,
- v tíme nie je pridelená rola Manažér kvality.

Čo by sme zlepšili:

- práca v tíme,
- aby bol počet projektov v škole menší,
- prideľovanie code review,
- vymazávanie vetiev, ktoré už boli spojené s hlavnou vetvou,

- štruktúra testov,
- časový manažment členov tímu,
- častejšie stand up-y,
- presné časové ohraničenie, dokedy majú byť napísané príspevky do stand up-ov,
- čítanie všetkých stand up-ových príspevkov všetkými členmi tímu.

Šprint 6 – Furby

Šiesty šprint bol prvý v letnom semestri a jeho trvanie bolo od 12. 2. 2018 do 26. 2018 a naplánovali sme si naň tieto úlohy:

1. Ukladanie hodnôt metrík (Min, Max, Between, Guaranteed) oddelene,
2. Upravenie RDF exportu,
3. Zobrazenie detailov porovnania control objectives,
4. Vytvorenie a zobrazenie prehľadu control questions,
5. Posudzovanie certifikačnej schémy,
6. Importovanie excelu s controls a control questions pre schému,
7. Neopakovanie sa atribútov v rámci jednej schémy,
8. Zautomatizovanie prepisu 1/5 - vytváranie control objectives,
9. Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík,
10. Vytvorenie abstraktu a priebežnej správy.

V rámci tohto šprintu sa nám nepodarilo dokončiť tri úlohy – Upravenie RDF exportu, Zautomatizovanie prepisu 1/5 - vytváranie control objectives a Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík. Dôvodom neúspechu druhej a tretej úlohy bolo zoznamovanie sa s technológiou Natural Language Processing, ktoré bolo náročnejšie, ako sme očakávali.

Retrospektíva

Čo sa nám páčilo:

- snaha člena tímu dokončiť ťažkú úlohu,
- člen tímu mal väčšinu práce hotovú už v prvom týždni šprintu,
- naučenie práce s excelom v Python-e,
- migrácie v aplikácii,
- člen tímu pomohol druhým,
- code reviews fungovali lepšie ako minulý semester,

Čo sa nám nepáčilo:

- rozloženie úloh,

- slabší šprint,
- zmena špecifikácie počas šprintu,
- požiadavky na špecifikáciu,
- komunikácia medzi členmi tímu,
- veľa práce,
- scrum master nevedel, ako má motivovať ľudí,
- nedostatočná spätná väzba členov tímu k článku na TP CUP,
- člen tímu nenašiel implementované riešenie pre parsovanie komplexných viet na jednoduché.

Čo by sme zlepšili:

- rozloženie úloh,
- špecifikáciu úloh,
- odhadovanie úloh,
- komunikáciu v tíme,
- spoluprácu,
- migrácie.

Šprint 7 – Game Boy

V rámci tohto šprintu, ktorý prebiehal od 26. 2. 2018 do 12. 3. 2018, sme pracovali na týchto úlohách:

1. Zautomatizovanie prepisu 1/5 - vytváranie control objectives,
2. Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík,
3. Zautomatizovanie prepisu 3/5 - vytváranie nových atribútov,
4. Zautomatizovanie prepisu 4/5 - vytváranie nových metrík,
5. Upravenie a odstraňovanie existujúcich schém, control-ov, control question-ov, control objective-ov,
6. Upravenie RDF exportu.

Všetky úlohy z tohto šprintu sme úspešne dokončili.

Retrospektíva

Čo sa nám páčilo:

- dokončenie úlohy s RDF export-om,
- člen tímu pomohol s code review,
- ochota člena tímu pomáhať s úlohami,

- vzájomné pomáhanie si,
- zistenie toho, čo robí dlhý skript v html.

Čo sa nám nepáčilo:

- dlhý skript v súbore html,
- code reviews sa robia príliš neskoro,
- nezvládla sa spojiť automatizácia prepisu,
- zlá komunikácia členov tímu o problémoch,
- dorábanie úloh v nedeľu v noci,
- nasadzovanie v priebehu stretnutia,
- nedokončenie úlohy, ktorá trvá už 4 týždne.

Čo by sme zlepšili:

- písanie skriptov nie do súborov html,
- spravenie code reviews skôr,
- pripravenie úloh na koncošprintové odovzdanie,
- skript v html,
- robenie úloh nie v nedeľu večer.
- komunikáciu, keď má člen tímu problém.

Šprint 8 – Hrkálka

Náplňou ôsmeho šprintu v trvaní od 12. 3. 2018 do 26. 3. 2018 boli tieto úlohy:

1. Dokončenie TP CUP článku,
2. Zautomatizovanie prepisu 5/5 - plná automatizácia,
3. Zautomatizovanie prepisu control questions - plná automatizácia,
4. Refaktorovanie databázy,
5. Naplnenie systému schémou CCM - Controls,
6. Upravenie fixtures.

Úlohu Naplnenie systému schémou CCM - Controls sa nám podarilo spraviť iba čiastočne, keďže to bola veľmi náročná úloha. Dokončenie úlohy Upravenie fixtures sme kvôli nepredvídateľným problémom museli presunúť do nasledujúceho šprintu. Ostatné úlohy sme úspešne dokončili a odovzdali.

Retrospektíva

Čo sa nám páčilo:

- pomoc členov tímu,
- člen tímu sa stal znalcom Natural Language Processing-u,
- členovia tímu pracovali na svojich úlohách,
- teambuilding,
- aspoň nejaká časť schémy CCM je prepísaná.

Čo sa nám nepáčilo:

- administratívna práca prepisovania schémy CCM,
- napĺňanie databázy údajmi,
- nekonzistencia v informáciách a dátach,
- vznikali nezhody v tom, kto ako čo robil pri prepisovaní schémy,
- slabá spätná väzba na TP CUP článok,
- neuploadovanie zápisníc načas,
- chaos v aplikácii,
- nedokončené úlohy.

Čo by sme zlepšili:

- vytvorenie vetvy develop,
- ujasnenie si detailov pred tým, ako začneme pracovať na úlohe,
- uploadovanie zápisníc načas,
- priraďovanie code review na začiatku šprintu,
- skonzistentnenie dát.

Šprint 9 – Igráček

Počas tohto šprintu od 26. 3. 2018 do 9. 4. 2018 sme robili tieto úlohy:

1. Upravenie generovania control objectives,
2. Vytvorenie prvého návrhu posteru na TP CUP,
3. Vytvorenie prezentácie tímu na TP CUP,
4. Zaregistrovanie novej služby,
5. Fixovanie bugov,
6. Upravenie fixtures,
7. Pripravenie ISO 27k a testovanie importu,
8. Opravenie existujúcich control objectives,
9. Prerobenie porovnania schém,
10. Vytvorenie staging branch.

Úlohy z tohto šprintu sme splnili všetky načas.

Retrospektíva

Čo sa nám páčilo:

- projekt sa hýbe dopredu,
- všetky úlohy boli na konci šprintu dokončené,
- práca člena tímu,
- opis control objective-ov,
- debata o motivácii,
- členovi tímu nebol pridelený code review,
- vedúci tímu pochválil Martina G.

Čo sa nám nepáčilo:

- agresívne debaty,
- účasť na TP CUP,
- príliš veľké staranie sa product owner-a do fungovania tímu,
- nedôvera product owner-a,
- rozloženie práce v tíme,
- tímový duch,
- nedostatočná motivácia niektorých členov tímu.

Čo by sme zlepšili:

- spôsob komunikácie,
- rozdelenie úloh,
- záujem o TP CUP.

Šprint 10 – Jojo

V rámci šprintu v trvaní od 9. 4. 2018 do 23. 4. 2018 sme pracovali na nasledujúcich úlohách:

1. Fixnutie konfliktu cookies,
2. Naplnenie systému schémou ISO 27k (tabuľkové),
3. Vylepšenie importu schémy,
4. Vylepšenie graficko-používateľského rozhrania,
5. Pripravenie stránky na prezentovanie na TP CUP,
6. Vytvorenie tlačidla Skúsim šťastie,
7. Pripravenie oviec na TP CUP,
8. Vytvorenie posteru TP CUP,

9. Implementovanie procesu revieweovania schémy,
10. Zmysluplné chybové stránky (4xx, 5xx).

Všetky úlohy sme úspešne načas odovzdali.

Retrospektíva

Čo sa nám páčilo:

- stihli sme spraviť všetky naplánované úlohy,
- že si člen tímu poriadne pozrel a pohral sa s CSS súborom,
- väčšina vecí z aplikácie funguje podľa predpokladov,
- výzor chybových stránok,
- úspešné zvládnutie účasti na TP CUP,
- origami ovečky,
- poster na TP CUP.

Čo sa nám nepáčilo:

- problémy člena tímu s PyCharm,
- že sme neboli v TOP 2 najlepších tímoch na TP CUP,
- neaktívna účasť niektorých členov tímu na TP CUP,
- neustále hádanie sa dvoch členov tímu,
- naladenie „koniec a hotovo“.

Čo by sme zlepšili:

- slovenskú stránku aplikácie (aby nepadala),
- naladenie členov tímu.

Šprint 11 – Koniec

Počas posledného šprintu od 23. 4. 2018 do 7. 5. 2018 sme mali naplánované tieto úlohy:

1. Upravenie homepage a upravenie dashboard-u schémy,
2. Znovuposielanie chýb do Rollbar-u,
3. Opravenie toho, že prepísanie ‘en’ na ’sk’ zhodí aplikáciu,
4. Zinteligentnenie javascript-u,
5. Zdokumentovanie finálneho projektu,
6. Finalizovanie projektu,
7. Diseminovanie výsledkov.

Dokončili sme všetky úlohy z tohto šprintu.

Retrospektíva

Čo sa nám páčilo:

- všetko sa stihlo,
- mali sme toho trochu menej,
- že je posledné stretnutie,
- člen tímu vnímal svoju priradenú úlohu ako upokojujúcu,
- členovi tímu sa páčilo, že nemal internet,
- vedúceho neporazilo, keď bola malá účasť na štvrtkovom stand-up stretnutí,
- že člen tímu spísal inštalačnú príručku a druhý nakreslil aktualizovaný dátový model a architektúru systému,
- aj keď sme nerobili priebežne, všetko sa stihlo.

Čo sa nám nepáčilo:

- suverénne to priebehovo bol najslabší šprint,
- že sa členovia tímu na začiatku šprintu tvárili, že toho majú veľa,
- pylint používalo málo ľudí,
- že niektorí členovia tímu mohli robiť viac, ale nestíhali.

Čo by sme zlepšili:

- písanie správ do štvrtkového stand-up.

Globálna retrospektíva ZS

Počas zimného semestra sa nám podarilo úspešne začať prácu na projekte. Oboznámili sme sa s technológiami a prostredím, a začali sme vývoj. Za čas jedného semestra sa nám podarilo implementovať základnú funkcionalitu, ktorá však ešte stále má svoje muchy. Museli sme si taktiež zvyknúť na agilný vývoj softvéru v tíme, aj keď niektorí členovia nášho tímu už s ním mali aspoň základné skúsenosti.

Medzi problémy, s ktorými sme bojovali a podarilo sa nám nájsť riešenie, patria:

- ohodnocovanie a rozdeľovanie úloh
  - Táto schopnosť sa v našom tíme prirodzene zlepšovala každým šprintom, ktorý sme absolvovali.
  - Viedli sme diskusie a spätne sme sa rozprávali o tom, či sa nám to podarilo zlepšiť.
- zodpovedné písanie poriadnej dokumentácie
  - Zo začiatku semestra sme mali mierny problém s dokumentáciou, keďže sme jej nevenovali dostatočnú pozornosť.
  - Zadefinovala sa teda metodika na dokumentáciu a začali sme dbať na to, aby dokumentácia vznikala naozaj zároveň s úlohami.
- komunikácia mimo stretnutí a na stretnutiach
  - Mimo stretnutí bola niekedy komunikácia nášho tímu slabšia.
  - Usilovali sme sa o jej zlepšenie či už vo virtuálnej podobe, ale aj formou teambuilding/u, napr. na Beánii a spoločných obedoch a posedeniach.
  - Na stretnutiach sme sa snažili nehádať sa, aj keď nie vždy sa nám to darí, keďže máme v tíme niekoľko silných osobností, ktoré sa niekedy nevedia dohodnúť. Nikdy sa však nejedná o závažné konflikty, ktoré by ohrozovali vzťahy v tíme či projekt celkovo.

Problémy, ktoré ešte stále riešime:

- pomalý rozbeh práce v prvom týždni šprintu
  - Tento problém pretrváva celý semester, a zatiaľ sa nám ho nepodarilo vyriešiť, aj keď niektoré jeho aspekty sa nám podarilo zlepšiť.
  - Keďže pracujeme v 2-týždňových šprintoch, tak počas prvého týždňa sa často spraví minimum úloh, a následne sa nestíhajú dokončiť v druhom týždni.
  - Ako riešenie zvažujeme 1-týždňové šprinty.
- zlý časový manažment členov tímu
  - Spôsobuje ho hlavne veľké množstvo iných školských projektov.
  - Snažíme sa prispôsobiť plánovanie šprintov podľa nadchádzajúcich projektov z iných predmetov.

    - Nie vždy sa nám podarí správne odhadnúť náročnosť projektov z iných predmetov.
- písanie automatických testov
    - Písanie automatických testov bola pre náš tím nová skúsenosť.
    - Ešte sa nám úplne nepodarilo spriateliť sa s touto technológiu a písaním týchto testov.
    - Bola zadefinovaná metodika, ktorá pomáha pri písaní týchto testov, a s postupom času sa táto naša schopnosť zlepšuje.
- prideľovanie code review
    - Zaznamenaný počas posledného šprintu, kedy sa nám nepodarilo stihnúť urobiť všetky potrebné code review.
    - Dovtedy sme si tieto úlohy prideľovali na báze dobrovoľnosti.
    - Zvažujeme riešenie pevného poradia, podľa ktorého by sa prideľovali code review.

Globálna retrospektíva LS

Počas letného semestra sme pokračovali v práci na projekte. Hneď v prvom šprinte tohto semestra sme začali s implementáciou nového modulu – Natural Language Processing (NLP). Zoznámenie sa s ním nám trvalo o niečo dlhšie, ako sme predpokladali, avšak počas priebehu semestra sa niektorí členovia, dalo by sa povedať, stali expertmi na tento modul, a prácu s ním ovládajú veľmi dobre. Podarilo sa nám implementovať rozšírenú kľúčovú funkcionalitu našej aplikácie. Svoje riešenie sme odprezentovali v rámci súťaže TP CUP. Na agilný vývoj sme oproti zimnému semestru už boli viac zvyknutí a niektoré problémy, ktoré sme mali v minulom semestri, sa nám podarilo úspešne vyriešiť. S niektorými sme však bojovali naďalej a vyskytlo sa aj zopár nových.

Medzi vyriešené problémy z minulého semestra patria:

- prideľovanie code review
    - Na začiatku letného semestra sme sa dohodli na prístupe prideľovania code review metódou round robin.
    - Taktiež sme znížili počet povinných code review jednej user story z dvoch na jeden.
    - Vytvorili sme zdieľanú tabuľku s menami členov tímu a code review sa prideľovali na základe toho, kto bol práve na rade.
    - Code review sa prideľovali pri vytvorení úlohy pre code review v TFS členom tímu zodpovedným za user story, v rámci ktorej bol code review potrebný.
    - Na základe tohto rozhodnutia bola aktualizovaná aj Metodika na tvorbu features a user stories.
- písanie automatických testov
    - Na základe Metodiky na tvorbu testov vytvorenej v minulom semestri, sme sa postupne zlepšovali v písaní automatických testov.
    - Podarilo sa nám dospieť do stavu, kedy písanie automatických testov nie je ponechané iba na Manažéra testovania.
    - Každý člen tímu sa snaží písať si automatické testy na svoj kód sám.

Problémy, ktoré počas letného semestra aj naďalej pretrvávali:

- pomalý rozbeh práce v prvom týždni šprintu
    - Navrhovanú myšlienku jednotýždňových šprintov sme neaplikovali.
    - Naďalej sa nám stávalo, že sa počas prvého týždňa šprintu spravilo iba veľmi malé množstvo úloh a následne sa na konci šprintu nestíhalo úlohy dokončiť.
    - Zhruba v strede semestra sme sa dostali až do stavu, kedy sa úlohy dokončovali iba pár minút pred koncošprintovým stretnutím alebo až počas jeho priebehu.
    - Tento problém sme adresovali aj v retrospektíve šprintu 7.

    - Túto hraničnú situáciu sa nám podarilo v ďalších šprintoch neopakovať, avšak problém pomalého rozbehu práce v prvom týždni šprintu pretrvával počas celého semestra.
- zlý časový manažment členov tímu
    - Stále sa nám nepodarilo dokázať si primerane rozdeliť svoj čas medzi všetky školské projekty, ktoré musíme spraviť počas priebehu jedného šprintu.
    - Občas podceníme zložitosť úlohy, ktorá nám je v rámci šprintu pridelená a nevymedzíme si na ňu dostatočne veľa času.
    - Snažíme sa však navzájom si pomáhať, teda ak niektorý z členov tímu má svoju úlohu už dokončenú, je k dispozícii na pomoc ostatným.

Nové problémy, ktoré sme identifikovali počas letného semestra:

- nedostatočný tímový duch
    - Niektorí členovia tímu si neprečítali, a tým pádom sa nijako nevyjadrili k článku písanému na TP CUP.
    - Názor niektorých členov tímu na účasť v tejto súťaži nebol pozitívny, čo sa prejavilo aj na celkovej nálade v tíme.
    - Iní členovia tímu sa aktívne nezapojili do prezentovania počas konania TP CUP a nechali túto činnosť na ostatných.
- nerovnomerné rozdelenie úloh s ohľadom na ich náročnosť medzi členov tímu
    - Počas tohto semestra sa viackrát stalo, že niektorí členovia tímu mali vo viacerých šprintoch po sebe pridelené tie najvyššie ohodnotené user stories.
    - Toto sa prejavovalo aj na počte odpracovaných hodín, ktorý mali títo členovia tímu oveľa vyšší ako ostatní.
    - S touto situáciou nebol spokojný product owner, avšak my ako tím sme si do istej miery dokázali obhájiť, že toto je spôsob, akým fungujeme.
    - Pri ďalšom prideľovaní úloh sme si však dávali väčší pozor na to, komu bude pridelená ako náročná úloha.

Metodiky

V tejto časti sa nachádza opis metodík a odkaz na jednotlivé metodiky, ktoré boli zadefinované počas práce na projekte.

Metodika na code review

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MCR.pdf

Metodika bola vytvorená, aby členovia tímu vedeli, ako vytvárať pull-requesty, kde člen tímu môže zrecenzovať kód a akými krokmi musí prejsť, aby mohol byť kód merge-nutý do hlavnej vetvy. Je to sprievodca pre kontrolu kódu.

Všeobecné pokyny

- Prijmite, že mnohé rozhodnutia o programovaní sú názory.
- Dobré otázky sa vyhýbajú subjektívnemu úsudku a vyhýbajú sa perspektíve autorov.
- Požiadajte o objasnenie, ak niečomu nechápete.
- Vyhnite sa selektívnemu vlastníctvu kódu. ("v mojom kóde atď")
- Vyhnite sa používaniu výrazov, ktoré by mohli byť považované za odkazy na osobné znaky. ("je to hlúpe"). Predpokladajme, že každý je inteligentný a rozumný.
- Buďte explicitní. Pamätajte, že ľudia nie vždy chápu vaše úmysly online.
- Buďte pokorní. ("Nie som si istý - poďme sa pozrieť.")
- Nepoužívajte hyperboly. ("vždy", "nikdy", "nekonečne", "nič")
- Nepoužívajte sarkasmus.
- Ak je príliš veľa komentárov "nerozumiem" alebo "alternatívne riešenie:", diskutujte synchrónne (napr. chat, zdieľanie obrazovky, osobne). Uverejnite komentár, ktorý zhrnie diskusiu.

Všeobecné pokyny pre autora kódu

- Buďte vďační za návrhy recenzenta.

- Neberte komentáre osobne, recenzuje sa kód nie Vy.
- Je ľahké nesprávne interpretovať spätnú väzbu. Ak sa kontrola zdá byť agresívna alebo nahnevaná alebo inak osobná, zvážte, či to bolo v zámere recenzenta a osobne požiadajte o objasnenie zámeru, ak je to možné.
- Predpokladajte najlepšie zámery z pripomienok recenzenta.
- Vysvetlite, prečo kód existuje.
- Extrahujte niektoré zmeny pre budúcnosť.
- Snažte sa pochopiť perspektívu recenzenta.
- Skúste odpovedať na každý komentár.
- Vykonajte merge, akonáhle sa budete cítiť istý, že kód je v poriadku.

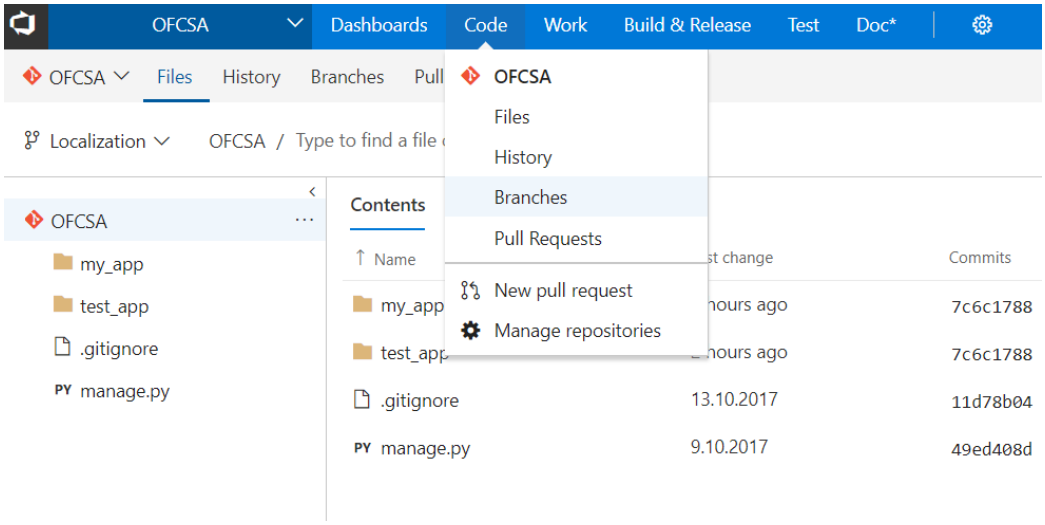
Všeobecné pokyny pre recenzenta

- Komunikujte, vypichnite silné stránky kódy a naopak aj tie slabé.
- Identifikujte spôsoby, ako je možné kód zjednodušiť.
- Ak sú diskusie príliš filozofické alebo akademické, presuňte diskusiu offline na pravidelné stand-up stretnutie. Medzitým nechajte autora urobiť konečné rozhodnutie o alternatívnych implementáciách.
- Ponúknite alternatívne implementácie. Predpokladajte však, že autor ich už zvažoval.
- Snažte sa pochopiť autorovu perspektívu.
- Ak už nemáme žiadne pripomienky, potvrďte merge s nejakou frázou, napr. "Pripravený na merge".

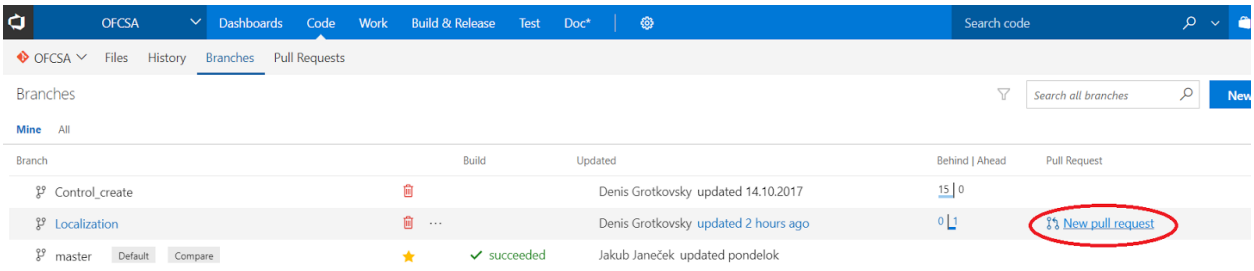
Pokyny pre autora

- po tom ako si vytvoril svoj kód, vykonaj commit a vytvor novú vetvu
- nasledovne urob push tejto vetvy na git
- na vytvorenie pull request-u z vytvorenej vetvy, postupuj takto:

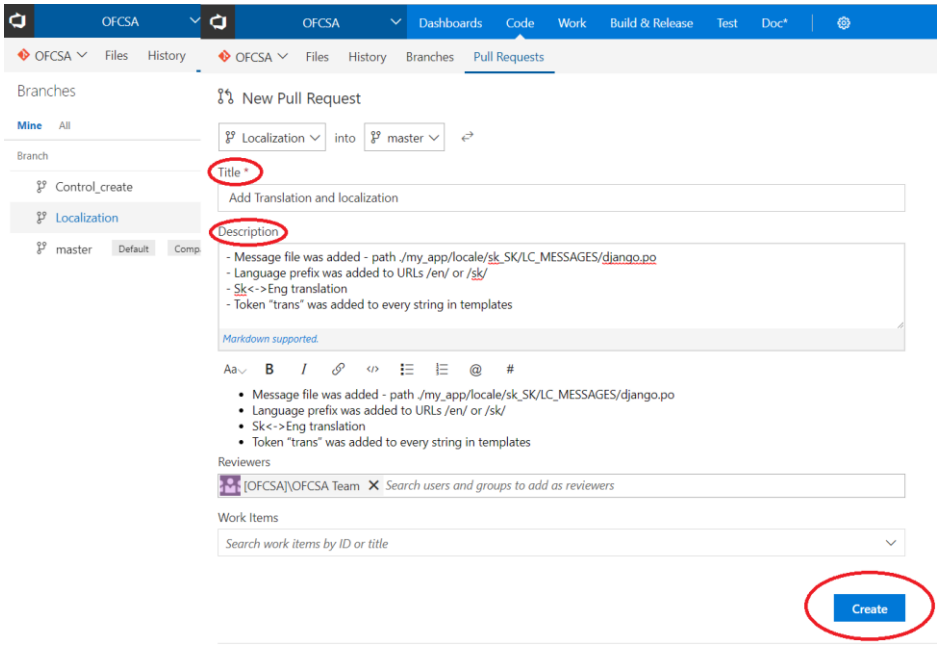
1. Na stránke TFS choď na Code -> Branches.



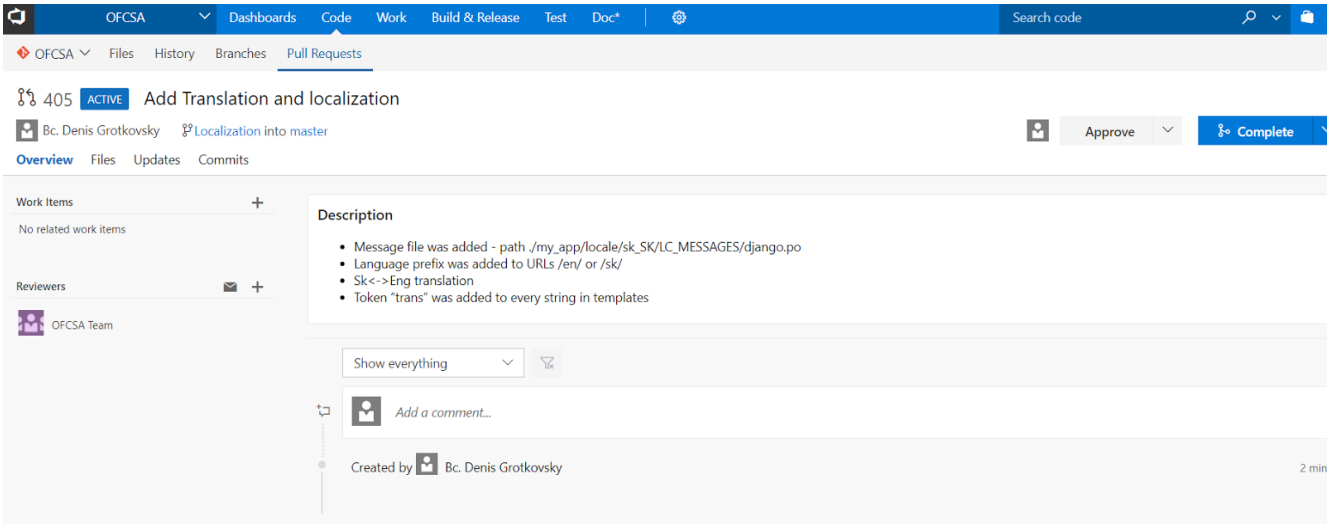
2. Tam nájdi vetvu, pre ktorú chceš vytvoriť pull request a stlač “New pull request”.



3. Vyplň “Title” a “Description” podľa metodiky na zdieľaný vývoj pomocou gitu a potvrď vytvorenie pull request-u stlačením “Create”.

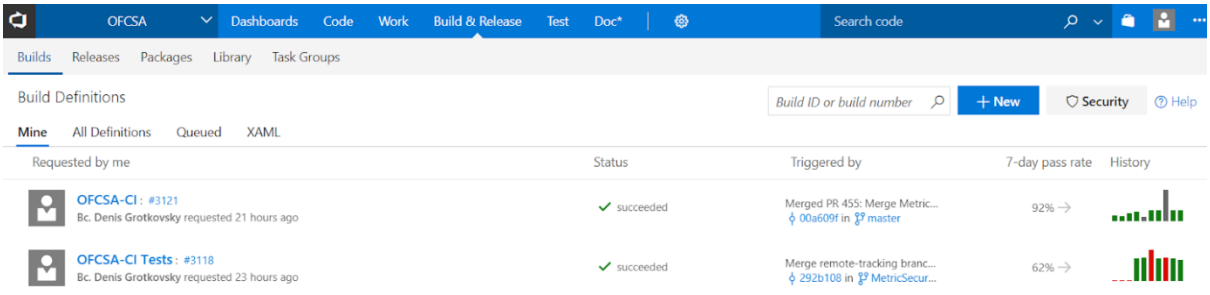


4. Po vytvorení ťa stránka presmeruje na “Overview” tvojho pull request-u. Na tejto stránke budú môcť tvoji recenzenti nahliadať na tvoj kód a písať komentáre. Komentáre sa píšu do “Overview”, alebo k jednotlivým riadkom kódu v možnosti “Files”. Na komentáre aktívne odpovedaj.



5. Ak nastane situácia, keď recenzent nájde nejakú chybu alebo nejasnosť v tvojom kóde a napíše ju do komentára, je potrebné túto chybu opraviť alebo odpísať na tento komentár a stlačiť tlačidlo “Resolve” alebo iné podľa situácie.

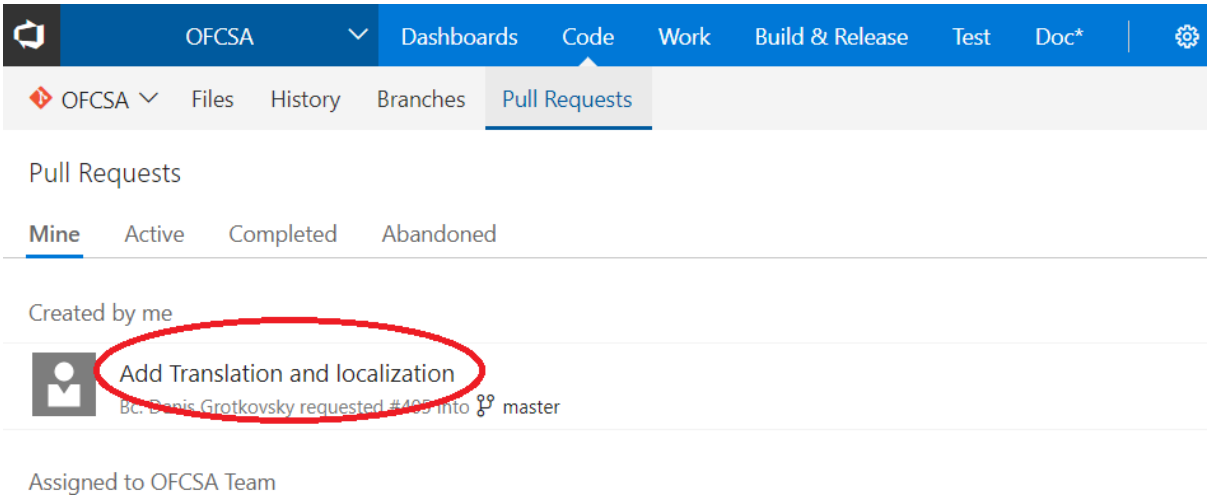
6. Tiež je potrebné skontrolovať či ti zbehol build. To nájdeš v “Build & Release” vo svojich build-och, kde klikneš na build svojho pull request-u. Build musí byť úspešný aby sa dal pull request schváliť.



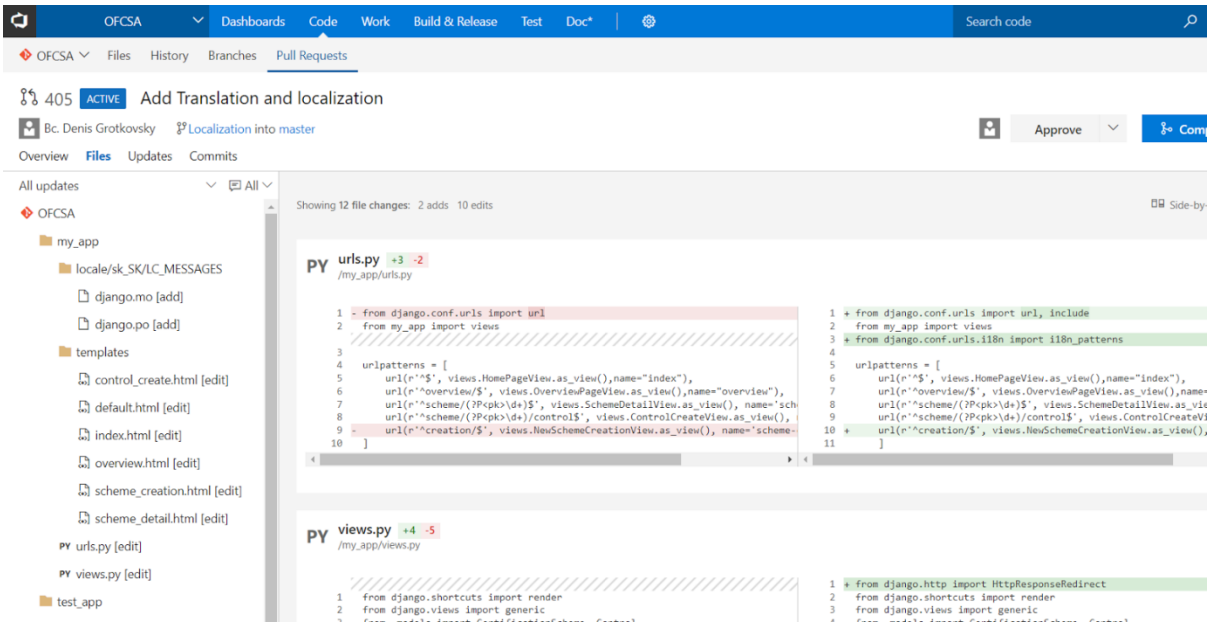
7. Po tom ako ti tvoj kód recenzent schváli a si s kódom spokojný, môžeš vykonať merge kódu pomocou tlačidla “Complete”.

Pokyny pre recenzenta

1. Po tom, ako ti bola pridelená úloha na code review, choď do Code -> Pull requests a nájdi názov pull request-u, ktorý ti bol pridelený.



2. V možnosti "Files" môžeš vidieť kód autora.

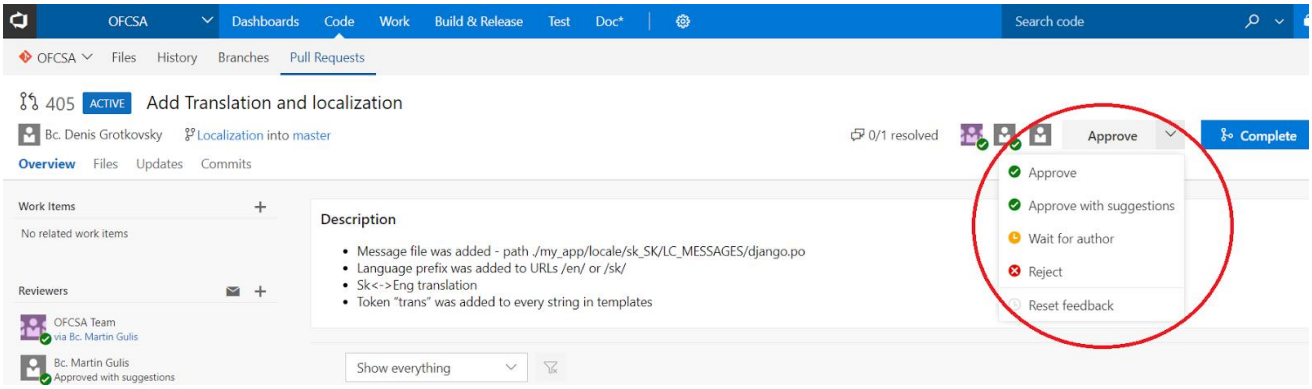


3. Po tom ako zrecenzuješ kód (pozri časť Čo je potrebné kontrolovať), možeš ho:

    a. **Approve** - schváliť pull request môžeš len vtedy, ak si prešiel všetky body v časti "Čo je potrebné kontrolovať" a všetko je v poriadku.

    b. **Reject** - zamietnuť pull request, ak obsahuje veľké chyby (zlá funkcionalita, zlé testy a pod), ktoré sa nedajú ľahko opraviť (dopísať chybný docstring a pod.).

4. Na malé chyby sa snaž upozorniť a napísať ich do komentárov alebo ku riadku kódu. Ak sa ti čokoľvek nezdá, tiež to napíš do komentárov alebo ku riadku kódu.



Čo je potrebné kontrolovať

1. Statická analýza
    a. Vymazal autor niečo, čo nemal?
    b. Je kód v súlade s metodikou na písanie kódu?
    c. Sú docstringy všade, kde majú byť?
1. Dynamická analýza
    a. Kód si stiahni a spusti
    b. Je jeho funkcionalita správna?
    c. Snaž sa otestovať funkcionalitu používateľskými úlohami.
1. Testy
    a. Zbehli všetky testy? (pozri bod 6 pri pokynoch pre autora)
    b. Testuje sa to, čo sa má testovať?

Ako písať komentáre a ako sa správať pri code review nájdeš tu:

https://github.com/thoughtbot/guides/tree/master/code-review

Doplnkové informácie

https://help.github.com/articles/reviewing-changes-in-pull-requests/

https://help.github.com/articles/about-pull-requests/

Metodika k lokalizácii a prekladu

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MLC.pdf

Metodika bola vytvorená kvôli pridaniu lokalizácie do aplikácie. Metodika opisuje, ako písať lokalizáciu a preklad v kóde. Pretože je program vytváraný pre Európsku úniu a jej štáty, je potrebný preklad stránky aj do iných jazykov. Základný jazyk, ktorým sa píše text v programe, je angličtina.

Ako písať lokalizáciu v TEMPLATE súboroch

Na začiatok všetkých html súborov, za prípadnú definíciu rozšírenia je potrebné vložiť tag {% load i18n %}. Toto je potrebné aj v tých, ktoré rozširujú html súbor, v ktorom je už tento tag použitý. Následne môžeme označiť reťazec na preklad.

Pôvodný:

<title>Hello world</title>

Nový:

<title>{% trans "Hello world" %}</title>

Použili sme tag {% trans %}. Tým sme povedali, že tento reťazec sa bude prekladať. Tento tag sa da využiť aj pri premenných:

<title>{% trans myvar %}</title>

Existuje aj tag {% blocktrans %}{% endblocktrans %}, ktorým je možné prekladať celé vety, obsahujúce reťazce aj premenné.

Toto sú len základy, odporúčam pozrieť dokumentáciu:

https://docs.djangoproject.com/en/1.11/topics/i18n/translation/#internationalization-in-template-code

Ako písať lokalizáciu v PYTHON súboroch

V python súboroch využijeme funkcie knižnice django.utils.translation, z ktorej použijeme funkciu ugettext.

from django.utils.translation import ugettext

Následne pri každom stringu, ktorý budeme chcieť prekladať využijeme ugettext():

var = ugettext("Hello world")

Pre zjednodušenie a zrýchlenie písania použijeme substitúciu:

from django.utils.translation import ugettext as _

A ďalej píšeme už len:

var = _("Hello world")

Opakujem a silne odporúčam pozrieť si dokumentáciu, pretože sú tam vysvetlené ďalšie veci ako pluralizácia, kontextové markeri atď, ktoré sa možno nevyužijú, ale budete vedieť, že sa vôbec niečo také dá využiť, a že to knižnica podporuje.

https://docs.djangoproject.com/en/1.11/topics/i18n/translation/#internationalization-in-python-code

Ako aktualizovať/vytvoriť message súbor

Vytvorenie a aktualizovanie message súboru je veľmi podobné, v niektorých krokoch identické. Message súbor je v tomto prípade aj vytvorený. Ale čo ak chceme pridať nový jazyk?

Vytvorenie:

1. Najprv je potrebné si stiahnuť *gettext*. Návod nájdete na tejto stránke pod nadpisom „gettext On Windows“:

https://djangobook.com/localization-create-language-files

2. Následne vytvoríme nový priečinok s názvom "locale" v priečinku "my_app".

3. Stlačíme ctrl+alt+r a do príkazového riadku napíšeme

makemessages -l sk_Sk

Namiesto sk_SK dajte požadovaný jazyk. Týmto sa vytvorí .po súbor, ktorý obsahuje všetky stringy na preklad:

msgid "Hello world"

msgstr ""

Všade je potrebné pridať preklad do požadovaného jazyka:

msgid "Hello world"

msgstr "Ahoj svet"

4. Ďalej je potrebné skompilovať .po súbor. Využijeme príkaz, ktorý zadáme do príkazového riadku:

compilemessages

To vytvorí .mo súbor. Následne by mal byť preklad viditeľný aj v aplikácii.

Aktualizovanie:

Podobné ako pri vytváraní, začíname bodom číslo 3.

Niečo bližšie k message súborom:

https://djangobook.com/localization-create-language-files

Metodika na štýl kódu – Python

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MSK.pdf

Metodika obsahuje pravidlá, ktorými sa má člen tímu riadiť pri písaní kódu. Na začiatku obsahovala len pravidlá pre jazyk Python, neskôr bola doplnená aj o pravidlá pre písanie HTML.

Python

V jazyku Python odsadenie ohraničuje bloky kódu (for, if,...)

```
if x == 3:
    print "we have 3"          # patrí do podmienky

print "we also have x"         # nepatrí do podmienky
```

Na odsadenie používame 4 medzery. Nepoužívame tab.

Ak máme na jednom riadku príliš dlhé volanie funkcie:

Správne:

```
foo = long_function_name(var_one, var_two,
                         var_three, var_four)
```

Nesprávne:

```
foo = long_function_name(var_one, var_two,
      var_three, var_four)
```

Pri podmienke if to môže vyzerať takto, ak je príliš dlhá na jeden riadok:

```
if (very_long_stuff and

   this_is_quite_long):

   print "we fulfilled both"
```

Vyššie uvedený príklad je dosť nečitateľný. Odporúčané riešenie je pridať riadok komentáru.

```
if (very_long_stuff and

   this_is_very_long_too):

   # If we fulfilled both

   print "we fulfilled both"
```

Ak vytvárame nejaký objekt, ktorého definícia je na viac riadkov, koncovú zátvorku uvádzame na samostatný riadok:

Správne:

```
list = [
   1, 2, 3,
   4, 5, 6,
   ]
```

Nesprávne:

```
list = [
   1, 2, 3,
   4, 5, 6, ]
```

Ak v kóde uvádzame nejaké matematické operácie, operátor uvádzame na nový riadok. Tento spôsob zápisu umožní ľahšie priradenie operandu k operátoru.

Správne:

```
result = (food

          + drinks
```

```
        + sweets
        - donations)
```

Nesprávne:

```
result = (food +
        Drinks +
        Sweets -
        donations)
```

Maximálna dĺžka riadku kódu je 79 znakov. Pre komentáre a docstring-y je maximálna dĺžka riadku 72 znakov.

Definície tried obaľujeme troma prázdnymi riadkami. Definície metód obaľujeme dvoma prázdnymi riadkami.

Premenné nie je potrebné oddeľovať prázdnymi riadkami, ale je možné to urobiť, ak chceme naznačiť logické rozdiely v skupinách premenných. Taktiež v kóde môžeme využiť prázdne riadky na oddelenie logických častí, čo zlepší zrozumiteľnosť kódu.

Import-y uprednostňujeme globálne oproti lokálnym. Mali by sa nachádzať na začiatku kódu. Jeden import na jeden riadok.

V jazyku Python nie je rozdiel v jednoduchých a dvojitých úvodzovkách pri uvádzaní reťazcov. My budeme používať pre uvádzanie reťazcov dvojité úvodzovky.

Príklad:

```
string = "This is a string"
```

Ak však reťazec obsahuje dvojité úvodzovky, uprednostňujeme použitie jednoduchých úvodzoviek oproti spätným lomítkam.

Správne:

```
string = 'Peter said: "Hello mom".'
```

Nesprávne:

```
string = "Peter said: \"Hello mom\"."
```

Snažíme sa písať kód tak, aby sme nemuseli písať komentáre. Teda chceme aby bol kód samovysvetľujúci. Ak však už komentáre píšeme, tak na samostatný riadok. Snažíme sa vyhnúť komentárom na rovnakých riadkoch s kódom.

Správne:

```
# x is radius

x = 3
```

Nesprávne:

```
x = 3   # x is radius
```

Dokumentačný reťazec, je taká štruktúra, ktorá nám umožní vygenerovať dokumentáciu pre projekt z reťazcov, ktoré v nej uvedieme. Dokumentačný reťazec by mal mať každý **modul, trieda, funkcia, metóda**. Dokumentačné reťazce píšeme vždy s dvojitými úvodzovkami. Ak je jednoriadkový, vyzerá nasledovne:

```
"""Documentation string on one line"""
```

Ak je na viac riadkov, vyzerá takto:

```
"""

This documentation string is not on one line.

It also has a second one.

"""
```

Príklad pre funkciu:

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    """


    if imag == 0.0 and real == 0.0:
    return complex_zero
    ...
```

Názvy premenných, funkcií, modulov a tried majú nasledovné pravidlá:

- funkcie, premenné a moduly: malé písmená, slová oddelené podčiarovníkom

  napr. -> certification_scheme_name

- triedy: veľké písmená na začiatku slov

  napr. -> CertificationScheme

Kód a názvy premenných vytvárame v anglickom jazyku. Taktiež komentáre píšeme po anglicky.

HTML

Každá HTML stránka bude obsahovať menu, ktoré sa nachádza v súbore default.html. Treba ho teda zahrnúť do vytváranej stránky, tým že na začiatok vytváraného html súboru vložíme: `{% extends "default.html" %}`

Stránka bude obsahovať navigáciu, pomocou ktorej sa používateľ dostane naspäť na predchádzajúcu stránku. Je teda potrebné vložiť blok breadcrumbs. Tento blok bude obsahovať minimálne domovskú stránku, na ktorú sa bude dať prekliknúť a vašu aktívnu stránku. Príklad:

```
{% block breadcrumbs %}

   <ol class="breadcrumb">

       <li class="breadcrumb-item"><a href="{% url 'index'%}">{%
trans "Homepage" %}</a></li>

       <li class="breadcrumb-item"><a href="{% url 'overview'
%}">{% trans "Certification scheme overview" %}</a></li>

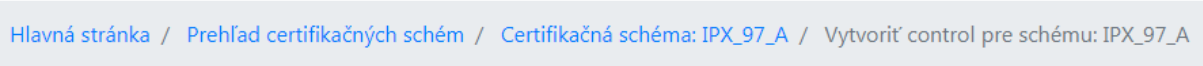
       <li class="breadcrumb-item active"><a href="/scheme/{{
scheme.pk }}"> {% trans "Certification scheme" %}: {{
scheme.scheme_name }}</a></li>

       <li class="breadcrumb-item active" aria-current="page">{%
trans "Control create for scheme" %}: {{ scheme.scheme_name
}}</li>

   </ol>

{% endblock %}
```

Na tomto príklade je možné vidieť, že ako prvá je domovská stránka, ďalej sa nachádza prehľad schém, ďalej konkrétna schéma a nakoniec vytváranie control-u pre vybranú schému. Vytváranie control-u je aktívna stránka, preto nie je možné na ňu kliknúť a nikam používateľa nepresmeruje. Čo ukážka kódu vytvorí, sa nachádza na Obrázku 1.



Obrázok 1. Navigácia na stránke.

Každá stránka by mala obsahovať nadpis s názvom danej stránky. Nadpis bude v tagu <h1>. Príklad je uvedený nižšie.

HTML kód musí byť správne odtabovaný a odriadkovaný. Vnútornejší tag je o jeden tab ďalej ako ten nad ním. To isté platí aj pre Django bloky. Obsah Django bloku je o jeden tab ďalej ako daný blok.

Príklad:

```
{% block content %}
   <h1>{% trans "Certification scheme" %}</h1>
   <table>
       <tr>
           <th>{% trans "Scheme's name" %}</th>
           <th>{% trans "Publisher" %}</th>
           <th>{% trans "Identifier" %}</th>
           <th>{% trans "Version" %}</th>
           <th>{% trans "Number of controls" %}</th>
       </tr>
       <a href="{% url 'index' %}">{% trans "Homepage" %}</a>
   </table>
{% endblock %}
```

Linky nepíšeme v html súboroch statické ale vo formáte **{% url %}**

Správne:

```
<a href="{% url 'create-control-objective' control.pk %}">
     {% trans "Add control objective" %}
</a>
```

Nesprávne:

```
<a href="/control/{{ control.pk }}/control_objective">
     {% trans "Add control objective" %}
</a>
```

Štýl pomocou css nastavujeme v html cez css súbory a nie priamo v kóde html. Ak chceme vložiť odkaz na css súbor, vkladáme ho do bloku na to určeného. Tento blok sa vkladá pred blok **breadcrumbs**.

Príkaz {% load static %} načíta potrebné súbory pre každý html template. Následne je možné vložiť odkazy na vlastné css súbory. Css súbory je potrebné umiestniť v štruktúre projektu do priečinku **static** a v ňom do priečinku **css**.

Príklad:

```
{% block stylesheets %}

    {% load static %}

    <link rel="stylesheet" href="{% static 'css/overview.css' %}">

{% endblock %}
```

Podobne ako štýl css sa vkladajú aj vlastné javascript súbory. Nepíšeme javascript priamo do html (existujú výnimky), vkladáme ho do bloku na to určeného. Tento blok sa vkladá na koniec html súboru. Javascript súbory je potrebné umiestniť v štruktúre projektu do priečinku **static** a v ňom do priečinku **javascript**.

Príklad:

```
{% block scripts %}

    <script src="{% static 'javascript/linking.js' %}"></script>

{% endblock %}
```

Django

Projekt v Django má niekoľko častí. Na najvyššej úrovni sa delí na **test_app** a **my_app**.

- test_app obsahuje globálne súbory pre celý projekt,
- my_app obsahuje súbory pre našu aplikáciu.

Priečinok **my_app** obsahuje priečinky a súbory.

- Priečinok **fixtures** obsahuje **.json** súbory, pomocou ktorých napĺňame databázu cvičnými údajmi.

- Priečinok **locale** obsahuje súbory využívané na preklad aplikácie do rôznych jazykov.
- Priečinok **migrations** obsahuje automaticky generované migrácie databázy (vytvorenie tabuliek).
- Priečinok **templates** obsahuje **.html** súbory reprezentujúce formuláre aplikácie.
- Súbor **forms.py** obsahuje definície tried, ktoré sa využívajú na generovanie obsahu html formulárov v prípadoch vytvárania alebo upravovania záznamov v databáze.
- Súbor **models.py** obsahuje definície entít pre našu aplikáciu (podklad pre tabuľky databázy).
- Súbor **tests.py** obsahuje testy pre aplikáciu.
- Súbor **urls.py** obsahuje definície url vzorov a ich preklad na controlleri formulárov.
- Súbor **views.py** obsahuje definície funkcií, ktoré slúžia ako controlleri pre aplikáciu.

Zdroje

https://www.python.org/dev/peps/pep-0008/

http://www.voidspace.org.uk/python/articles/python_style_guide.shtml#standards-and-style

Metodika na tvorbu features a user stories

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MFU.pdf

Metodika opisuje, akým spôsobom sa vytvárajú Features, User Stories a úlohy. Predpisuje tiež počet potrebných úloh, ktoré musia byť vytvorené pre code review v rámci každej User Story.

Feature

Feature predstavuje skupinu User Stories, ktoré spolu súvisia. Je to väčšia ucelená časť funkcionality vytváraného systému, ktorá je pre biznis dôležitá. Na to, aby bolo možné odhadnúť, ako dlho implementácia Feature potrvá, musí byť najskôr rozdelená na menšie časti – User Stories – pomocou ktorých dokážeme od zúčastnených strán dostať spätnú väzbu, aby sme zistili, či sa vývoj uberá správnym smerom.

User Story

- je požiadavka, ktorá jasne popisuje kto, čo a prečo to potrebuje,

- píšeme z pohľadu čo najkonkrétnejšie určeného používateľa,
- je ľahko manažovateľná, ľahko sa dá určiť, ako dlho bude trvať jej riešenie,
- sa dá realizovať v krátkom časovom intervale (3-5 dní) a sledovať jej priebeh,
- má akceptačné kritériá, na základe ktorých sa dá vyhodnotiť, či bola dokončená,
- predstavuje pre používateľa hodnotnú časť funkcionality systému,
- neobsahuje zbytočné slová, ktoré jej nepridávajú podstatnú informačnú hodnotu.

Názov User Story začína slovesným podstatným menom. Na odhad toho, ako dlho bude trvať implementácia User Story, používame Story Points, ktorých počet je určený po tímovej diskusii. Ak má niektorá User Story priveľa Story Points, snažíme sa rozdeliť ju na viac menších častí. User Stories sú ďalej rozčlenené na úlohy (Tasks). Na rozdiel od User Story, na jednej úlohe zvyčajne pracuje iba jeden človek, ktorý vykonáva jeden typ práce (dizajnovanie, programovanie, dokumentovanie...).

User Story sa postupne nachádza v piatich rôznych stavoch. Tieto sú:

1. New – vytvorená Product Owner-om, neohodnotená tímom,
2. Approved – ohodnotená tímom pomocou Story Points,
3. Committed – tím sa zaväzuje, že User Story vykoná,
4. Resolved – vyriešená,
5. Done – skontrolovaná Product Owner-om a vyhlásená za dokončenú.

Po dokončení všetkých úloh v rámci User Story člen tímu, ktorý je zodpovedný za túto User Story, ju presunie v časti Backlog Items zo stĺpca Commited do stĺpca Resolved. Týmto dáva na vedomie Product Owner-ovi, že je daná User Story vyriešená a pripravená na odovzdanie.

Task

Úlohy vyplývajúce z User Story vytvára člen tímu, ktorému je User Story priradená pri plánovaní šprintu. Každá úloha má názov, ktorý začína slovesom v neurčitku a môže mať opis, ktorý bližšie určuje, čo sa má v rámci nej vykonať. Jeden z členov tímu si vytvorenú úlohu priradí (alebo mu je priradená) vyplnením časti Assigned To. Člen tímu, ktorému je úloha priradená, odhadne počtom hodín, ako dlho mu potrvá spravenie úlohy, a tento údaj zapíše do časti Original Estimate a Remaining Work.

Súčasťou každej User Story, v rámci ktorej vznikla nová alebo bola upravená väčšia súčasť projektu, je úloha pre Code review. Táto úloha je pri jej vytvorení priradená členovi tímu, ktorý je na rade podľa vytvorenej tabuľky pre code review. Samozrejme musí byť dodržané, že code reviewer nepracoval na žiadnej inej úlohe v rámci danej User Story. Úlohám pre Code review sa pri ich vytváraní v častiach Original Estimate a Remaining Work priradí 1 hodina.

Keď člen tímu začne na úlohe pracovať, zmení jej stav z To Do na In Progress. Počas práce na úlohe zaznačuje do časti Completed Work počet hodín, ktoré už strávil na plnení úlohy, pričom v časti Remaining Work aktualizuje počet hodín, ktoré mu ešte zostávajú do dokončenia danej úlohy. Keď je úloha dokončená, zmení jej stav z In Progress na Done.

Zdroje

https://scrum.sk/blog/2017/06/12/priklad-user-story/

https://www.symphonious.net/2006/05/18/features-vs-stories/

Metodika na tvorbu testov

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MTT.pdf

Metodika vznika pre členov tímu, aby sa vedelo, čo a ako sa bude testovať. Zahŕňa vysvetlenie testovania pre Python aplikáciu v našom projekte. S postupným vývojom testov a možnosťami testovania sa rozrástla aj o návody pre testovanie jednotlivých častí aplikácie. Tiež obsahuje návod na inštaláciu potrebných programov na testovanie a spúšťanie testov.

Typy testov

Jednotkové testy

- Izolované testy, ktoré testujú jednu konkrétnu funkciu.
- Jednoduchšie sa píšu.
- Čím viac bude jednotkových testov, tým menej treba integračných testov.

Integračné testy

- Väčšie testy, ktoré sa sústreďujú na správanie a testujú celú aplikáciu.
- Testujú, ako spolu jednotlivé komponenty pracujú.

Čo sa má testovať

Testujú sa všetky aspekty kódu, ktoré sa môžu pokaziť alebo nemusia fungovať správne. Čím viac testov, tým lepšie. Netestujú sa žiadne knižnice ani funkcie poskytnuté od Python alebo Django, pretože tie si oni testujú sami. Testuje sa teda iba vlastný kód.

Štruktúra testov

Testy sa nachádzajú v adresári ‘tests‘. V prípade, že je mnoho testov v jednej z úrovní tohto adresára, vytvoria sa podadresáre. Súbory s testami majú tvar ‘test_*.py‘. Jednotlivé testy sú rozdelené do súborov na základe toho, čo testujú (napr. ‘test_model_control.py‘, ‘test_view_control_detail.py‘, ...). Je vhodné oddeliť jednotkové testy od integračných. Príklad adresárovej štruktúry je nasledovný:

/tests/

__init__.py

/integration_tests/

__init__.py

test_html_control_detail.py

test_html_scheme_detail.py

/unit_tests/

__init__.py

/model/

__init__.py

test_model_control.py

/view/

__init__.py

test_view_control_detail.py

Tvorba testov

Na tvorbu testovacích tried sa využíva trieda ‘TestCase‘ z knižnice django.test. Táto knižnica vychádza zo štandardnej knižnice unittest pre python.

Triedy na testovanie majú tvar ‘*TestCase(TestCase)’. Metódy na testovanie majú tvar ’test_*()‘.

Na konfiguráciu testov sa môžu využiť metódy triedy TestCase (ak nie sú potrebné, netreba ich písať), najznámejšie sú

- setUp()
  - zavolá sa pred každou testovacou metódou,
  - slúži na vytvorenie objektov alebo inú prípravu pre ostatné testy v triede,
  - každá metóda dostane novú verziu objektu,
- tearDown()
  - zavolá sa po každej testovacej metóde,
  - slúži na “upratanie“ po teste,
- setUpClass()
  - zavolá sa iba raz za celú triedu, pred zavolaním ostatných metód,
  - slúži na vytvorenie objektov, ktoré sa nebudú v žiadnych testovacích metódach meniť,
  - argumentom metódy je trieda podľa konvencií nazývaná ‘cls’,
  - metóda musí byť označená ako ‘classmethod‘:
    - ```
      @classmethod
      def setUpClass(cls):
          ...
      ```
- tearDownClass()
  - zavolá sa iba raz za celú triedu, po zavolaní ostatných metód,
  - slúži na “upratanie“ po testovaní celej triedy,
  - argumentom metódy je trieda podľa konvencií nazývaná ‘cls’,
  - metóda musí byť označená ako ‘classmethod‘:
      ```

        - @classmethod
          def tearDownClass(cls):
              ...

    - o ďalších metódach je možné sa dozvedieť viac v dokumentácii: https://docs.python.org/3/library/unittest.html

Obsahom testu je zavolanie metódy ‘assert*()‘. Zoznam týchto metód je pomerne rozsiahly, preferuje sa nepoužívať iba ‘assertFalse()‘ a ‘assertTrue()‘, ale používať čo najvhodnejšiu metódu vzhľadom na test. Zoznam všetkých metód je možné nájsť v dokumentácii: https://docs.python.org/3/library/unittest.html

Preferuje sa čo najmenej “Asserts per Test“, ideálne “One Assert per Test“. Jednotlivé testy majú byť od seba nezávislé. Poradie vykonania testov nie je v réžii autora testov.

Príklad testovacej triedy:

```
from django.test import TestCase


class YourTestCase(TestCase):

    def setUp(self):

        #Setup run before every test method.

        pass


    def tearDown(self):

        #Clean up run after every test method.

        pass


    def test_something_that_will_pass(self):

        self.assertFalse(False)
```

Testovacia databáza

Nie je potrebné vytvárať novú databázu na testovanie. Django si vytvorí na začiatku testovania vlastnú (z existujúcej databázy definovanej zo súboru settings.py s prístupom do nej zo súboru

settings.ini). Táto databáza bude obsahovať iba čisté tabuľky bez údajov a počas testovania sa štandardne používať. Po testovaní ju následne sám odstráni. Do databázy je možné ukladať záznamy štandartnými metódami ako pri písaní netestového kódu.

Majme triedu:

```
from django.db import models


class CertificationScheme(models.Model):
    """Model for certification scheme"""
    scheme_name = models.CharField(max_length=100)
    identifier = models.CharField(max_length=50)
    version = models.CharField(max_length=10)
    publisher = models.CharField(max_length=50)
```

Vytváranie tejto triedy je napríklad nasledovné:

```
CertificationScheme.objects.create(scheme_name='test_name',
                                   identifier='test_identifier',
                                   version='test_ver',
                                   publisher='test_publisher')
```

V tomto prípade si však treba dať pozor na atribúty, ktoré zvyčajne databáza vytvára sama (napr. pk alebo id). Autor testu sa nemôže spoliehať, že budú vychádzať vždy z prednastavených hodnôt (napr. pk=1), pretože databáza svoje záznamy priebežne odstraňuje po každej testovacej triede, ale neresetuje svoje interné počítadlá (napr. v rámci jednej testovacej triedy bude vytvorený prvý záznam s pk=1, ale v rámci ďalšej testovacej triedy bude vytvorený záznam s pk=2, pritom záznam s pk=1 nebude existovať).

Lepším, prehľadnejším a odporúčaným spôsobom je využívať fixtures. Sú to súbory vo formáte Json, ktorými sa naplní databáza v rámci testovacej triedy pred spustením ľubovoľnej testovacej metódy. Testovacie fixtures vytvárame v adresári ‘/fixtures/tests/‘. Súbory s testovacími fixtures majú tvar ‘test_*.json‘.

Príklad fixtures zo súboru ‘/fixtures/tests/test_schemes.json‘:

```
[
  {
```

```
  "model": "my_app.CertificationScheme",
  "pk": 1,
  "fields": {
    "scheme_name": "Profi",
    "identifier": "PF",
    "version": "1.0",
    "publisher": "FIIT"
  }
 }
]
```

Pre načítanie fixtures do testovacej triedy stačí využiť preddefinované pole ‘fixtures‘ a zadať názov súboru (‘fixtures’ je pole, preto je možné zadávať ľubovoľný počet súborov). Príklad načítania fixtures:

```
from django.test import TestCase
from my_app.models import CertificationScheme

class CertificationSchemeTestCase(TestCase):
    """Test case for certification scheme in models."""

    # This will load fixtures from file 'fixtures/tests/test_schemes.json'
    fixtures = ['tests/test_schemes']

    def test_fixture(self):
        “”"Test method for fixture test."""”
        scheme = CertificationScheme.objects.get(pk='1')
        self.assertIsInstance(scheme, CertificationScheme)
```

Testy pre jednotlivé časti aplikácie

Ako testovať jednotlivé časti aplikácie je možné nájsť napríklad v tejto dokumentácii: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing

Každá časť aplikácie sa testuje inak. V tejto metodike budú uvedené tie základné.

Testovanie models

Testovanie je zvyčajne jednotkové a závisí od toho, čo je v testovanej triede uvedené. Zvyčajne sa však testuje každá metóda v testovanej triede, a to aspoň raz (ak existujú nejaké špeciálne prípady alebo scenáre, tak všetky). Jedna testovacia metóda by nemala testovať viac ako jednu testovanú metódu.

Majme nasledujúcu triedu:

```
from django.db import models


class CertificationScheme(models.Model):
        """Model for certification scheme"""
        scheme_name = models.CharField(max_length=100)
        identifier = models.CharField(max_length=50)
        version = models.CharField(max_length=10)
        publisher = models.CharField(max_length=50)

        def __str__(self):
                return self.scheme_name

        def temp(self):
                return self.identifier
```

Príklad testu môže vyzerať takto:

```
from django.test import TestCase

from my_app.models import CertificationScheme
```

```
class CertificationSchemeTestCase(TestCase):
    """Test case for certification scheme in models."""
    fixtures = ['tests/test_schemes']

    def test_creation(self):
        """Test method for object creation."""
        scheme = CertificationScheme.objects.get(pk='1')
        self.assertIsInstance(scheme, CertificationScheme)

    def test_str(self):
        """Test method for str method."""
        scheme = CertificationScheme.objects.get(pk='1')
        self.assertIsInstance(scheme, CertificationScheme)
        self.assertEqual(scheme.__str__(), scheme.scheme_name)

    def test_temp(self):
        """Test method for temp method."""
        scheme = CertificationScheme.objects.get(pk='1')
        self.assertIsInstance(scheme, CertificationScheme)
        self.assertEqual(scheme.temp(), scheme.identifier)
```

Metóda ‘test_creation(self)‘ testuje, či pri vytváraní a získaní objektu nedošlo k chybe (nie je to testovanie funkcie Pythonu, neimportujeme na to žiadnu knižnicu a my sme túto triedu vytvorili, takže to môžeme tiež overiť).

Posledné dve metódy ‘test_str(self)‘ a ‘test_temp(self)‘ testujú metódy v testovanej triede ‘__str__(self)‘ a ‘temp(self)‘. Pred tým ale skontrolujú, či pri vrátení objektu nedošlo k chybe (nevadí, že toto testuje aj prvá testovacia metóda, pretože testovacie metódy majú byť od seba nezávislé – pri písaní týchto dvoch testov treba predpokladať, že metóda ‘test_creation(self)‘ nemusí existovať).

Testovanie views

Testovanie views je možné jednotkovými testami. Vytvorili sme pomocnú testovaciu triedu pre prihlásenie používateľa, pretože ju budú potrebovať viaceré testovacie triedy:

```
from django.test import TestCase

from django.contrib.auth.models import User


class SetUpTestCase(TestCase):

    """Test case for creating user.Also offers login method for user."""

    @classmethod

    def setUpClass(cls):

        """setUpClass method for creating user."""

        super(SetUpTestCase, cls).setUpClass()

        test_user = User.objects.create_user(username='testuser',

                                             password='12345')

    def login(self):

        """Login method for user."""

        self.client.login(username='testuser', password='12345')
```

Majme nasledujúcu triedu pre view:

```
class SchemeDetailView(generic.DetailView):

    @method_decorator(login_required)

    def get(self, request, **kwargs):

        scheme = CertificationScheme.objects.get(pk=kwargs['pk'])

        info = {'scheme': scheme}

        return render(request, 'scheme_detail.html', context=info)
```

Príklad testu môže byť nasledovný:

```
from django.core.urlresolvers import reverse
from my_app.tests.unit_tests.test_setup import SetUpTestCase

class SchemeDetailTestCase(SetUpTestCase):
    """Test case for scheme detail in views."""
    fixtures = ['tests/test_schemes']

    def test_get_successful(self):
        """Test method for successful get method."""
        self.login()
        resp = self.client.get(reverse('scheme-detail', args=['1']))
        self.assertEqual(resp.status_code, 200)

    def test_get_login_required(self):
        """Test method for failed get method because login is required"""
        resp = self.client.get(reverse('scheme-detail', args=['1']))
        self.assertEqual(resp.status_code, 302)

    def test_get_uses_correct_template(self):
        """Test method for correct template usage."""
        self.login()
        resp = self.client.get(reverse('scheme-detail', args=['1']))
        self.assertEqual(resp.status_code, 200)
        self.assertTemplateUsed(resp, 'scheme_detail.html')
```

Trieda ‘SetUpTestCase‘ nám zabezpečí vytvorenie používateľa a poskytuje metódu na jedno prihlásenie.

Metóda ‘test_get_successful(self)‘ testuje, či je stránka správne načítaná po prihlásení (status_code=200).

Metóda ‘test_get_login_required(self)‘ testuje, či stránka správne vyžaduje prihlásenie používateľa (status_code=302).

Metóda ‘test_get_uses_correct_template(self)‘ testuje, či stránka využíva správny template po prihlásení. Pred tým ale skontroluje, či je status_code=200 (nevadí, že toto testuje aj prvá testovacia metóda, pretože testovacie metódy majú byť od seba nezávislé – pri písaní týchto dvoch testov treba predpokladať, že metóda ‘test_get_successful(self)‘ nemusí existovať).

Testovanie htmls

Tieto testy štruktúrujeme na základe html súborov, ktoré predstavujú obrazovky pre používateľa. Preto sa jedná o integračné testy. Vytvorili sme pomocnú testovaciu triedu pre zapnutie browsera a prihlásenie používateľa, pretože ich budú potrebovať viaceré testovacie triedy:

```
import os


from django.contrib.staticfiles.testing import StaticLiveServerTestCase

from django.core.urlresolvers import reverse

from django.test import override_settings

from django.contrib.auth.models import User

from pyvirtualdisplay import Display

from selenium import webdriver


@override_settings(LANGUAGE_CODE="en", LANGUAGES=(("en", "English"),))

class SetUpTestCase(StaticLiveServerTestCase):

    """Test case for setup methods for integration testing."""

    fixtures = ["tests/test_schemes"]

    user_username = "testuser"

    user_email = "testuser@test.com"

    user_password = "12345"
```

```
@classmethod
def setUpClass(cls):
        """setUpClass method for opening browser."""
        super(SetUpTestCase, cls).setUpClass()
        if os.name == "posix":
                cls.display = Display(visible=0, size=(1366, 720))
                cls.display.start()
        cls.driver = webdriver.Chrome()
        cls.driver.set_window_size(1366, 720)


@classmethod
def tearDownClass(cls):
        """setUpClass method for closing browser."""
        cls.driver.quit()
        if os.name == "posix":
                cls.display.popen.kill()
                super(SetUpTestCase, cls).tearDownClass()


def setUp(self):
        """setUp method for creating user. """
        super(SetUpTestCase, self).setUp()
        User.objects.create_user(username=self.user_username,
                                 email=self.user_email,
                                 password=self.user_password)
```

```
    def insert_login_credentials(self, username, password):
        """Insert login credentials, click and test session cookie."""
        self.driver.find_element_by_id("id_username").clear()
        self.driver.find_element_by_id("id_username").send_keys(username)

        self.driver.find_element_by_id("id_password").clear()
        self.driver.find_element_by_id("id_password").send_keys(password)

        self.driver.find_element_by_xpath(
            '//button[contains(., "Login")]').click()
        self.assertIsNotNone(self.driver.get_cookie("sessionid"))

    def login_user(self, url):
        """Login method for basic user."""
        self.driver.get("%s%s" % (self.live_server_url, url))
        self.insert_login_credentials(self.user_username,
                                      self.user_password)
```

Majme nasledujúcu časť urls:

```
urlpatterns = [
    url(r'^$', views.HomePageView.as_view(), name="index"),
    url(r'^page/$', views.TestPageView.as_view(), name="page"),
    url(r'^accounts/', include('django.contrib.auth.urls')
]
```

Podstránka ‘page‘ využíva html s časťou:

```
<a href="{% url 'index' %}">{% trans "Homepage" %}</a>
```

Príklad integračného testovanie by mohol byť takýto:

```
from django.core.urlresolvers import reverse
from .test_setup import SetUpTestCase

class MyTestCase(SetUpTestCase):
    """Test case."""

    def test_to_homepage(self):
        """Test method for redirect to homepage"""
        self.login_user(reverse('login'))
        self.assertEqual("%s%s" % (self.live_server_url,
                                   reverse("page")
        self.driver.find_element_by_xpath(
            '//a[contains(., "Homepage")]').click()
        self.assertEqual('%s%s' % (self.live_server_url,
                                   reverse('index')),
        self.driver.current_url)
```

Trieda ‘SetUpTestCase‘ nám zabezpečí správne otvorenie/zatvorenie browsera aj vytvorenie používateľa a poskytuje nám metódy pre prihlásenie.

Metóda ‘test_to_homepage(self)‘ testuje odkaz v html. Je vhodné sa vyhýbať písaniu konkrétnej URL vo forme stringu, preto sa využíva pomocná premenná ‘self.live_server_url’ a metóda ‘reverse()‘. Táto metóda z názvu v jej argumente vytvorí URL príponou (argumentom metódy je názov zadefinovaný v ’urls.py‘). Najskôr prebehne prihlásenie používateľa na adrese s názvom ‘login‘. Následne sa dostaneme na stránku s názvom ‘page’. Vyhľadá sa element pomocou xpath výrazu a vykoná sa kliknutie. Nakoniec sa overí, či prebehlo presmerovanie url.

Spúšťanie testov (na Windowse)

Je nutné mať nainštalované:

- pyvirtualdisplay – cez command line príkazom ‘pip install pyvirtualdisplay‘
- selenium – cez command line príkazom ‘pip install selenium‘
- Google Chrome – stiahnuť zo stránky

  (https://www.google.com/chrome/browser/desktop/index.html)
- Chromedriver – zo stránky stiahnuť verziu na win32, v zip súbore sa nachádza chromedriver.exe

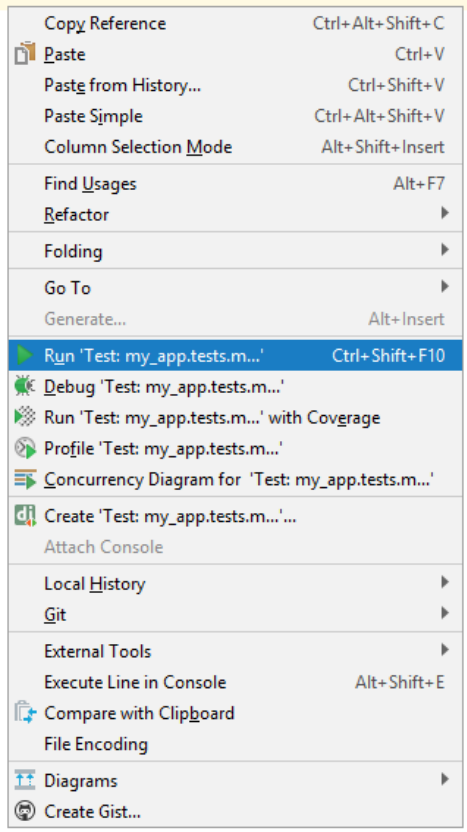
  (https://chromedriver.storage.googleapis.com/index.html?path=2.33/)

Následne je potrebné pridať cestu k priečinku so súborom chromedriver.exe medzi systémové premenné prostredia (do ‘Path’) a reštartovať počítač.

Testy celej aplikácie je možné spúšťať príkazom ’./manage.py test‘ nad adresárom so zdrojovým kódom z command line alebo z konzoly PyCharm. Pre konzolu v PyCharm po stlačení ctrl+alt+R stačí napísať iba “test‘:

```
manage.py@OFCSA > test
```

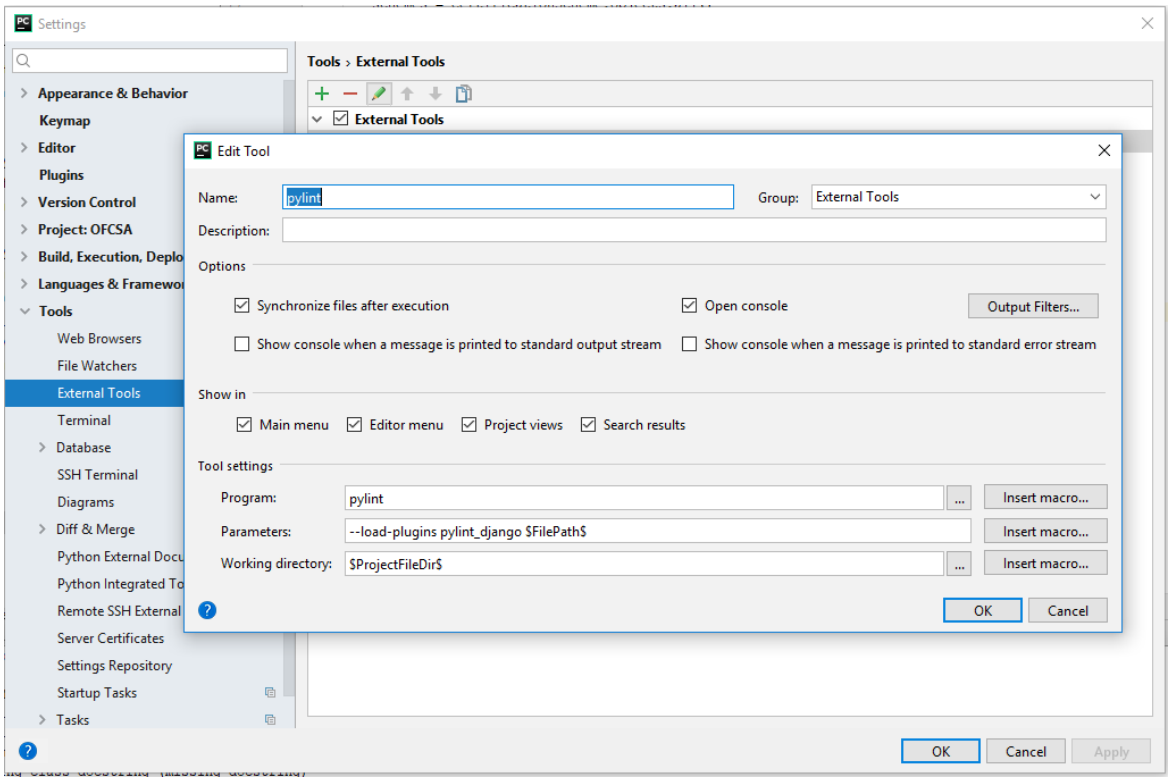
Jednotlivé testy z PyCharm je možné spúšť aj pravým kliknutím na konkrétnu triedu alebo metódu, spustia sa iba testy na základe miesta kliknutia:



Statická analýza kódu

Na statickú analýzu kódu sa používa pylint s pluginom django. Stiahnuť je to možné cez command line príkazom ‘pip install pylint-django‘. Na konfiguráciu využívame súbor ‘.pylintrc’.

Do PyCharm je možné pylint pridať cez File->Settings->Tools->External Tools->Add(zelený “+“) a takto nakonfigurovať:



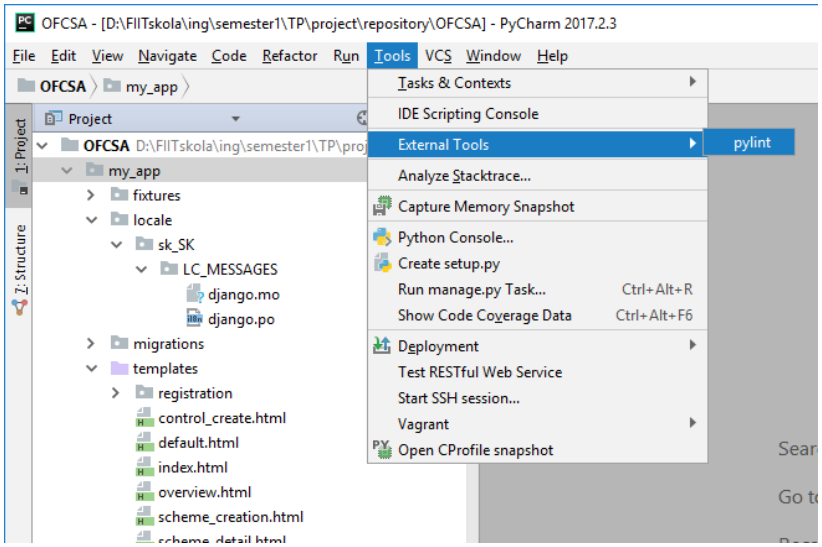
Name: pylint

Program: pylint

Arguments: --load-plugins pylint_django $FilePath$

Working directory: $ProjectFileDir$

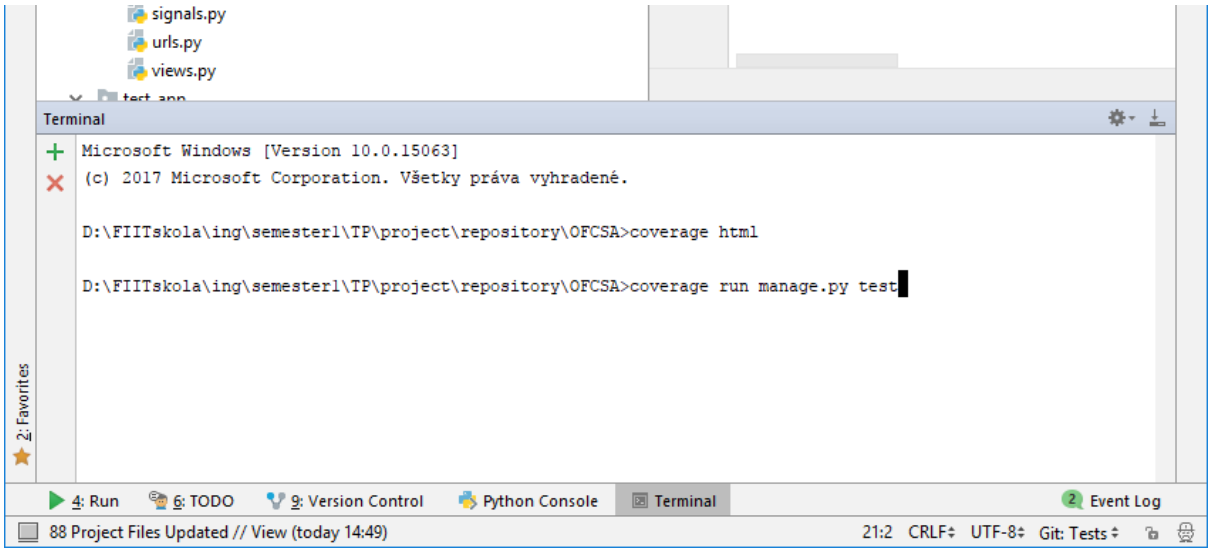
Ak bol pylint takto nakonfigurovaný, spúšťanie je nasledovné (otestujú sa všetky ‘*.py‘ súbory v adresári a podadresároch – závisí od označeného adresára, v tomto príklade ‘my_app’):



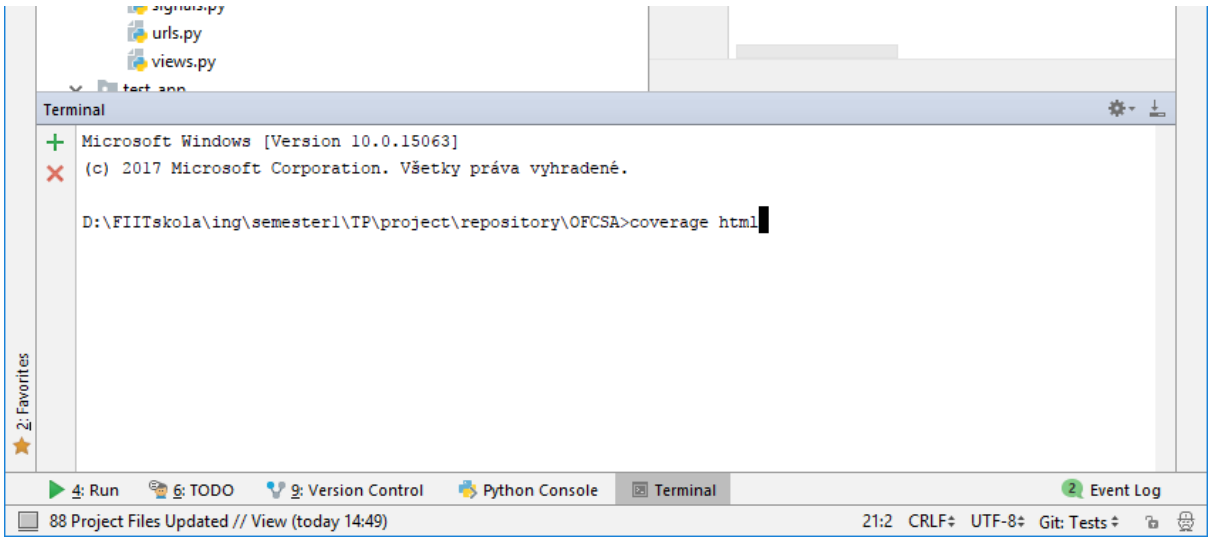
Pokrytie testami

Pri písaní kódu sa môže stať, že niektoré časti autor zabudne otestovať. Preto ako pomôcku využívame ’coverage’. Stiahnuť je to možné cez command line príkazom ‘pip install coverage‘. Na konfiguráciu využívame súbor ‘.coveragerc’.

Spúšťanie nie je možné cez django priamo cez ‘manage.py’. Je nutné využiť command line alebo Terminal z PyCharm. Zapnutie coverage prebieha nad adresárom so zdrojovým kódom príkazom ’coverage run manage.py test‘. Pre Terminal v PyCharm:



Následne prebehnú všetky testy. Následne je možné vygenerovať výstup, najlepšie ako html príkazom ’coverage html‘:



Výstupom je priečinok htmlcov v adresári so zdrojovými súbormi. Po otvorení základného vygenerovaného súboru ‘htmlcov/index.html’ je možné sa preklikávať v prehliadači a vidieť konkrétne zdrojové súbory s vyznačenými neotestovanými riadkami.

Zdroje

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing

https://docs.djangoproject.com/en/1.11/topics/testing/

https://realpython.com/blog/python/testing-in-django-part-1-best-practices-and-examples/

https://docs.python.org/3/library/unittest.html

http://django-testing-docs.readthedocs.io/en/latest/fixtures.html

Metodika na zdieľaný vývoj pomocou gitu

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MZV.pdf

Metodika vznikla, aby členovia tímu vedeli, čo a ako sa bude vkladať na git. Obsahuje vysvetlenia, ako vkladať časti kódu na git, do akých vetiev, ako aj kto musí riešiť konflikty v kóde na gite.

Rozdelenie Gitu

Štruktúra hlavného repozitára

Vytvorený je jeden hlavný repozitár tzv. origin, ktorý obsahuje jednu hlavnú vetvu a vedľajšie vetvy. Hlavnou vetvou je Master vetva a vedľajšie vetvy.

Master Vetva

Je to hlavná vetva. Obsahuje produkčnú verziu aplikácie. Vždy obsahuje stabilnú a funkčnú verziu aplikácie. Do tejto vetvy sa môže pridávať nový kód len z vedľajších vetiev, ktoré boli schválené po code review, ale to len v špecialnych prípadoch. Inak sa pridáva iba zo Staging vetvy.

Staging Vetva

Kópia master vetvy po poslednom šprinte. Najskôr sa všetky features mergujú do Staging a následne sa Staging vetva nahrá do mastra.

Vedľajšie vetvy

Tieto vetvy slúžia pre vývojárov na pushovanie nových častí kódu. Táto vetva sa nachádza tiež v repozitári origin a po schválení sa pushne do vetvy Master.

Nová vetva sa sa vytvorí v nástroji TFS, kde sa jej dá jednoznačný názov v angličtine a určia sa úlohy, ktoré sa do vetvy budú commitovať, čo je v nástroji umožnené. Názov každej vetvy sa začína veľkým písmenom. V prípade viacerých slov budú začiatočné písmená slov písané veľkým písmenom, teda používa sa Camel case. Názov vetvy by mal vyjadrovať jej obsah a jej názov je písaný v angličtine.

Príklad na názov vetvy na tvorbu menu:

MenuCreation

Zlúčenie vedľajšej vetvy do Master/Staging vetvy

Po daní pull request-u na vedľajšiu vetvu, je potrebné túto vetvu schváliť. Schvaľovať môže iba niekto, kto daný kód nevytvoril. Code review pre vetvu musí urobiť jeden člen tímu, ktorý je určený podľa pravidla definovaného v Metodike na tvorbu features a user stories. Až po schválení pull request-u daným členom tímu, môže byť pull request schválený a dokončený. Pull requesty by mali smerovať do Staging vetvy. Do mastra je z vedľajších vetiev dovolené mergovať len vo výnimočných situáciách.

Commitovanie

Commitovať treba ucelené a ideálne funkčné celky kódu.

Commit message

V prvom riadku commit správy by sa mal nachádzať názov commit-u. Tento názov by mal jasne vyjadrovať čo obsahuje commit. Pod nadpisom sa nachádza krátky opis commitu, ktorý by mal byť stručný a jasne vyjadrovať obsah zmien a pridaného kódu. Správy commitov sa píšu v anglickom jazyku.

Príklad:

Bootstrap addition

I created css for all forms and add bootstrap.

Revertovanie zmien

Na odstránenie súboru z commitu, bez odstránenia zmien sa použije reset Head nazov_súboru.

Ak je potrebné zabudnúť zmeny, použije sa príkaz checkout názov_súboru.

V prípade chyby v commite nie je potrebné vytvoriť ďalší commit, ale treba použiť príkaz commit amend, ktorý prepíše posledný commit.

Push

Po vytvorenom commite je potrebné zmeny vložiť do vytvorenej vetvy. Toto sa urobí príkazom Push.

Stiahnutie si novej verzie Master/Staging vetvy

Keď sa spojí vedľajšia vetva a hlavná vetva, teda vznikne nová verzia Master/Staging vetvy, je potrebné, aby všetci pracujúci na projekte mali najnovšiu verziu Master/Staging vetvy. Toto urobia pomocou príkazu Update. V prípade, že medzi Master/Staging verziou kódu a verziou kódu, ktorý robí Update nie je konflikt, vetva sa automaticky sama obnoví. V druhom prípade je potrebné vyriešiť konflikty a urobiť merge.

Príkaz Update obsahuje pull request a následne urobí merge projektu. V tomto príkaze je možné nastaviť aj to či sa bude vykonávať Merge alebo nie. V našom prípade sa bude robiť Merge. To znamená, že vyberieme možnosť Merge v okne, ktoré nám zobrazí po výbere príkazu Update project.

Mergeovanie

V prípade, že má niekto rozpracovanú zmenu a do vetvy master bola medzičasom schválená nová verzia, je potrebné svoje zmeny najprv commitnúť a až potom je možné získať novú verziu. Konflikty v kóde si musí vyriešiť autor kódu.

Pull Request

Title - jasný nadpis pre pull request

Description - ako prvé bude opis, čo sa v pull requeste nachádza. Ďalšie budú pridané súbory a následne zmeny, ktoré sa urobili v ostatných už existujúcich súboroch. Tieto opisy nemusia byt dlhé, stačí jasný, krátky opis. Ďalej sa v pull requeste dá určiť, na aké úlohy je daná vetva zameraná. Túto časť je potrebné vyplniť. Všetko sa píše v angličtine.

Príklad:

Sidebar menu creation and bootstrap addition

I added bootstrap to all forms and create new sidebar menu with css.

Added folder static for css files.

Added file default.css for menu stylesheet.

Modified deafult.html- added sidebar menu.

Pomoc pri zadávaní príkazov

https://www.jetbrains.com/help/pycharm/using-git-integration.html

Zdroje

http://labss2.fiit.stuba.sk/TeamProject/2014/team17is-si/metodika_git.pdf

https://git-scm.com/docs

Metodika na písanie dokumentácie

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MPD.pdf

Metodika vznikla, aby členovia tímu vedeli, čo majú kedy vložiť do dokumentácie a aký formát majú používať.

Úprava dokumentu

Pri písaní dokumentácie sa držte jednoduchých pravidiel:

1. Typ písma používajte Times New Roman.
2. Pre normálny text použite veľkosť 12 typ Times New Roman.
3. Pre nadpisy použite veľkosť 20, modrou farbou.
4. Neprekračujte 3 úrovne nadpisov.
5. Obrázky a tabuľky centrujte na stred, každý obrázok a tabuľku označte a stručne popíšte.
6. Text nikdy nepíšte vedľa obrázkov alebo tabuliek, vždy pod alebo nad.
7. Tabuľky musia byť zrozumiteľné a musí byť jasné, čo polia obsahujú (rozumné nadpisy stĺpcov).
8. Na názov stĺpcu v tabuľke použite bold.
9. Použite rozumnú veľkosť a kvalitu obrázku tak, aby bol čitateľný.

Ak je potrebná v dokumentácii úvodná strana, mala by obsahovať:

1. názov školy a fakulty,
2. názov tímového projektu,
3. skratka pre názov tímového projektu,
4. určenie aká dokumentácia to je (dokumentácia k riadeniu a pod.),
5. vedúci tímu: Meno Priezvisko,
6. členovia tímu: Mená a priezviská oddelené čiarkou,
7. školský rok.

Ak je potrebný obsah:

- generuje sa automaticky,
- nadpis: Obsah

Ukážka jednotlivých komponentov spomínaných vyššie aj s nastaveniami potrebných štýlov je v dokumente *Príklady komponentov*.

Čo a kam písať?

Pridanie funkcionality

Po úspešnom merge do master vetvy nezabudnite dopísať pridanú funkcionalitu do dokumentácie inžinierskeho diela, konkrétne do časti Implementácia a ak je funkcionalita rozsiahla, je nutné dopísať aj používateľskú príručku a technickú dokumentáciu. Inšpirovať sa môžete predošlými dokumentáciami.

Po ukončení tímového stretnutia je nutné vygenerovať tabuľku s úlohami a vložiť ju do Dokumentácie k riadeniu. Toto by mal vykonať zapisovateľ z príslušného cvičenia.

Dokumentácia inžinierskeho diela a Dokumentácia k riadeniu sa nachádzajú v priečinku Dokumenty.

Metodiky

Pri vytváraní novej metodiky alebo úprave starých metodík je dôležité dodržiavať pravidlá pre úpravu dokumentácie.

Vzorový dokument pre metodiku je možné nájsť v priečinku Metodiky->Metodika k dokumentácii v súbore *Metodika šablóna.*

Zápisnice

V zápisniciach sa namiesto písma Times New Romans používa písmo Arial. Každá zápisnica musí obsahovať nadpis, čas a dátum konania a zoznam prítomných osôb (mená neprítomných osôb ostanú zapísané, ale preškrtnú sa, napr. ~~Janko Hraško~~). Stav plnenia úloh z predchádzajúceho stretnutia a Úlohy do ďalšieho stretnutia sú vygenerované do tabuliek v nástroji TFS. Stručný priebeh a Výsledky stretnutia zapíše študent, ktorý má na danú zápisnicu na stretnutí na starosti.

Vzorový dokument je možné nájsť v priečinku Zápisnice v súbore *Šablóna.*

Používateľské príručky

Každá používateľská príručka musí obsahovať svoj názov. Pri pokynoch, ktoré má používateľ vykonať na stránke, je vhodné vložiť obrázok, na ktorom je graficky zobrazené, čo má používateľ robiť (kam má kliknúť). Každý obrázok má mať aj popis.

Vzorový dokument je možné nájsť v priečinku Metodiky->Metodika k dokumentácii v súbore *Používateľská príručka šablóna.*

Retrospektíva

Retrospektívu stačí písať do txt súboru. Mala by obsahovať čo sa páčilo, nepáčilo a čo by zmenil, každého kto sa zúčastnil na ukončení sprintu.

Vzorový dokument je možné nájsť v priečinku Zápisnice v súbore *Retrospektíva šablóna.*

Metodika na manažment chýb

https://team12-17.studenti.fiit.stuba.sk/doc/ls/MMC.pdf

Metodika bola vytvorená na určenie pravidiel a správania sa k chybám. V metodike je pokrytý celý životný cyklus chyby, od jej hlásenia na Slack až po jej uzatvorenie.

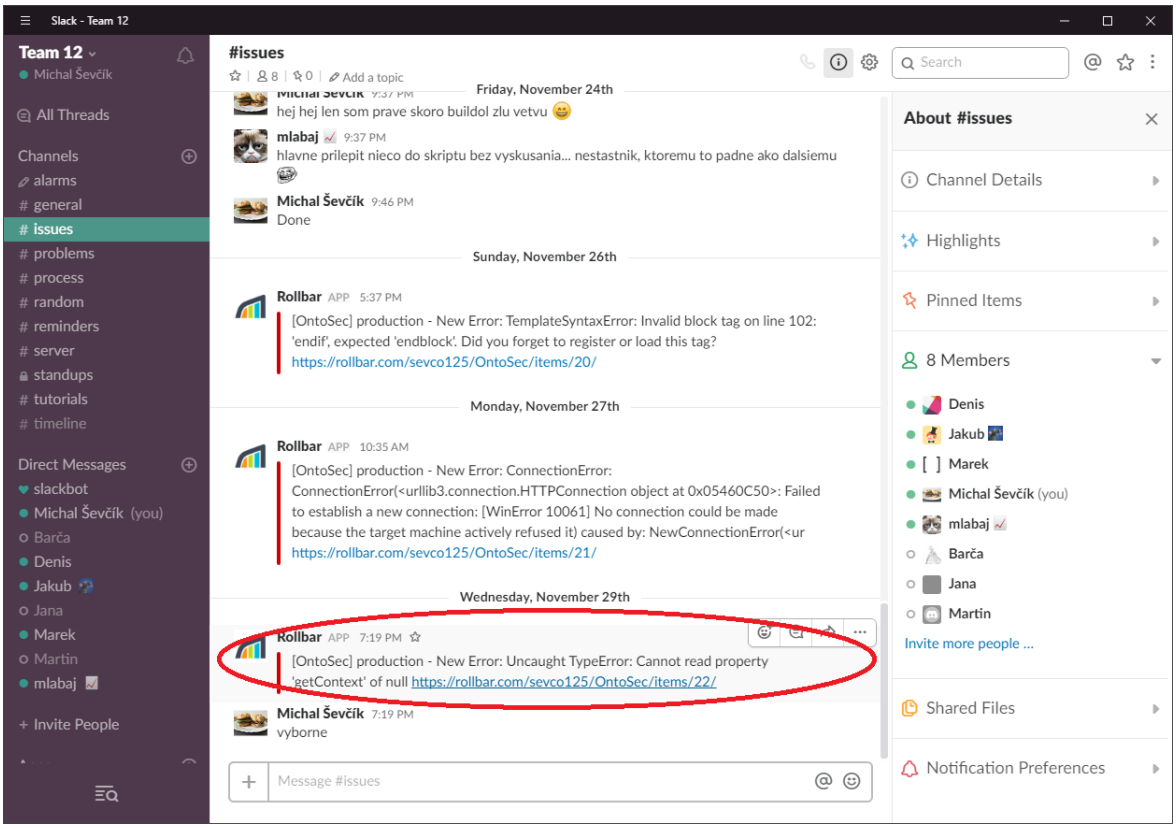
Na zachytávanie chýb je použitý systém rollbar. Prístup do tohto systému by mali mať všetci členovia tímu. Prihlasovacie údaje im boli poslané na e-mailovú adresu. V prípade že tieto údaje člen tímu stratil, kontaktuje autora tejto metodiky.

Ďalej je potrebný prístup do TFS. Ak niekto nemá prístup do TFS, musí to riešiť s kompetentnými osobami.

Ak máte prístup do Rollbar a tak isto máte prístup do TFS, tak spĺňate všetky požiadavky na to, aby ste sa mohli riadiť touto metodikou.

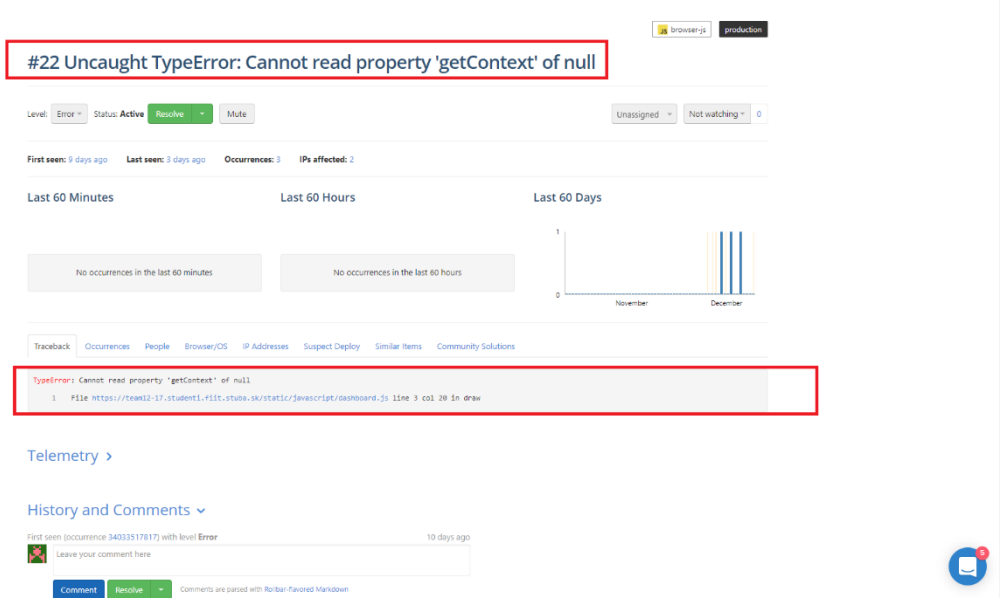
Životný cyklus chyby

Ak na serveri alebo v javascript-e v našej aplikácii nastane chyba, systém Rollbar ju zachytí a notifikuje nás pomocou Slack-u.



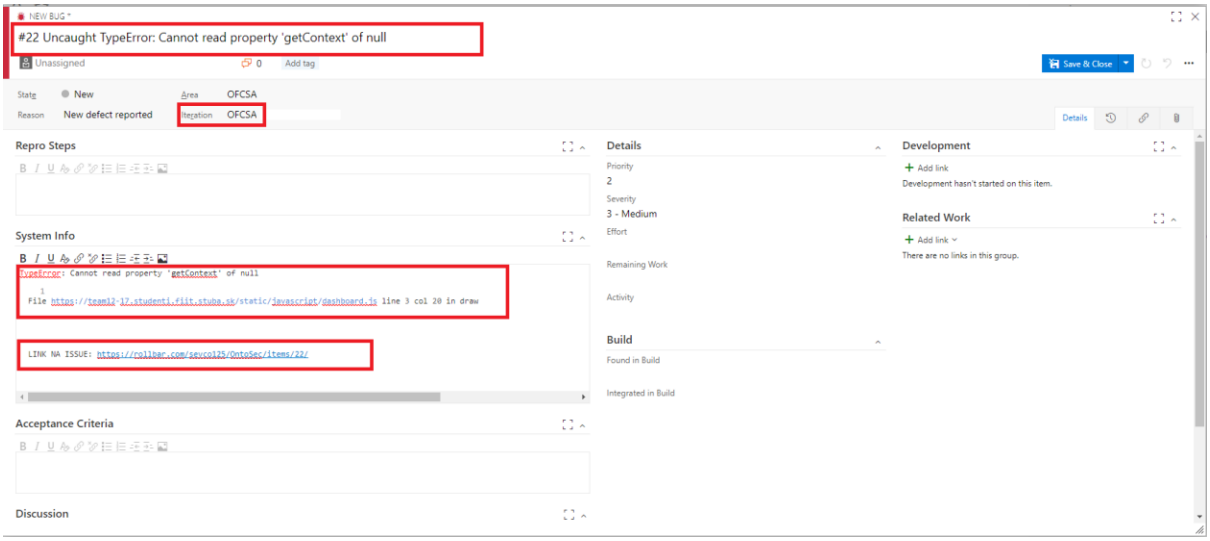
Následne je nutné aby niekto, zvyčajne Správca servera, ale nie je to limitované iba na neho, prepísal túto chybu do TFS ako Bug.

Najskôr rozklikneme link, ktorý nám prišiel v Slack notifikácii. Otvorí sa nasledujúce okno:



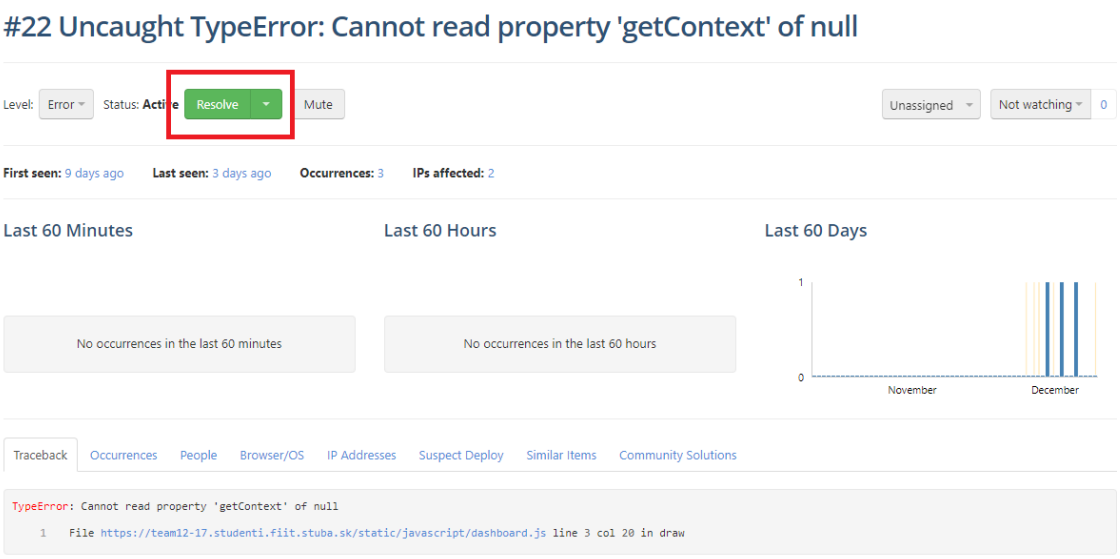
Kľúčové veci sú označené v červených obdĺžnikoch.

Otvoríme si TFS, prihlásime sa, v hornej navigačnej lište zvolíme Work a klikneme na Bug. Tu vyplníme údaje nasledovne:



Následne bug uložíme. Toto pridá Bug medzi Backlog items.

Po tom čo sa bug niekomu priradí a opraví, je potrebné ho odstrániť z Rollbar-u. To sa urobí tak, že sa klikne na LINK NA ISSUE a v systéme sa klikne na resolve:



Tým je pokrytý celý životný cyklus chyby.

Evidencia úloh

V tejto časti sa nachádza evidencia úloh, na ktorých sme pracovali počas jednotlivých šprintov. V každom šprinte je uvedený stav splnenia úloh po prvom aj po druhom týždni šprintu.

Šprint 1 – Autíčko

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6296	Vytvorenie novej certifikačnej schémy	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6295	Zobrazenie prehľadu certifikačných schém	Product Backlog Item	Bc. Jakub Janecek	Committed	
6294	Zobrazenie certifikačnej schémy	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6297	Vytvorenie control-u certifikačnej schémy	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
6299	Nainštalovanie servera	Product Backlog Item	Bc. Marek Vlha	Done	
6300	Vytvorenie webovej stránky tímu	Product Backlog Item	Bc. Michal Sevcik	Committed	
6301	Pripravenie aplikačného servera	Product Backlog Item	Bc. Michal Sevcik	Done	
6302	Vytvorenie metodiky na tvorbu features a user stories	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6303	Vytvorenie metodiky na tvorbu testov	Product Backlog Item	Bc. Martin Gulis	Committed	
6304	Vytvorenie metodiky na štýl kódu	Product Backlog Item	Bc. Jakub Janecek	Committed	
6307	Vytvorenie metodiky na zdieľaný vývoj pomocou gitu	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6305	Vytvorenie metodiky na code review	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6289	Prihlásenie administrátora	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	

6306	Vytvorenie automatického testovania	Product Backlog Item	Bc. Martin Gulis	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6296	Vytvorenie novej certifikačnej schémy	Product Backlog Item	Bc. Barbora Ungerova	Done	
6295	Zobrazenie prehľadu certifikačných schém	Product Backlog Item	Bc. Jakub Janecek	Done	
6294	Zobrazenie certifikačnej schémy	Product Backlog Item	Bc. Barbora Ungerova	Done	
6297	Vytvorenie control-u certifikačnej schémy	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6299	Nainštalovanie servera	Product Backlog Item	Bc. Marek Vlha	Done	
6300	Vytvorenie webovej stránky tímu	Product Backlog Item	Bc. Michal Sevcik	Done	
6301	Pripravenie aplikačného servera	Product Backlog Item	Bc. Michal Sevcik	Done	
6302	Vytvorenie metodiky na tvorbu features a user stories	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6303	Vytvorenie metodiky na tvorbu testov	Product Backlog Item	Bc. Martin Gulis	Done	
6304	Vytvorenie metodiky na štýl kódu	Product Backlog Item	Bc. Jakub Janecek	Done	
6307	Vytvorenie metodiky na zdieľaný vývoj pomocou gitu	Product Backlog Item	Bc. Barbora Ungerova	Done	
6305	Vytvorenie metodiky na code review	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6289	Prihlásenie administrátora	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	

Šprint 2 – Bábika

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6289	Prihlásenie administrátora	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
6528	Aktualizovanie metodík	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6306	Vytvorenie automatického testovania	Product Backlog Item	Bc. Martin Gulis	Committed	
6529	Vytvorenie automatického nasadzovania	Product Backlog Item	Bc. Michal Sevcik	Committed	
6535	Vytvorenie a zobrazenie prehľadu control objective-ov	Product Backlog Item	Bc. Jakub Janecek	Committed	
6534	Vytvorenie rozloženia a menu	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6530	Prihlásenie do TP Cupu	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
6533	Preloženie aplikácie do angličtiny	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
6531	Vytvorenie metodiky na dokumentáciu	Product Backlog Item	Bc. Marek Vlha	Committed	
6290	Vytváranie ďalších používateľských kont	Product Backlog Item	Bc. Marek Vlha	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6289	Prihlásenie administrátora	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6528	Aktualizovanie metodík	Product Backlog Item	Bc. Barbora Ungerova	Done	
6306	Vytvorenie automatického testovania	Product Backlog Item	Bc. Martin Gulis	Done	
6529	Vytvorenie automatického nasadzovania	Product Backlog Item	Bc. Michal Sevcik	Done	

6535	Vytvorenie a zobrazenie prehľadu control objective-ov	Product Backlog Item	Bc. Jakub Janecek	Done	
6534	Vytvorenie rozloženia a menu	Product Backlog Item	Bc. Barbora Ungerova	Done	
6530	Prihlásenie do TP Cupu	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6533	Preloženie aplikácie do angličtiny	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6531	Vytvorenie metodiky na dokumentáciu	Product Backlog Item	Bc. Marek Vlha	Done	
6290	Vytváranie ďalších používateľských kont	Product Backlog Item	Bc. Marek Vlha	Done	

Šprint 3 – Céčka

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6804	Refaktorovanie dokumentácie	Product Backlog Item	Bc. Marek Vlha	Committed	
6291	Zmena hesla používateľa	Product Backlog Item	Bc. Marek Vlha	Committed	
6292	Obnova hesla	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
6716	Zaznamenávanie novopridaných metrík	Product Backlog Item	Bc. Jakub Janecek	Committed	
6715	Full-text vyhľadávanie existujúcich metrík	Product Backlog Item	Bc. Michal Sevcik	Committed	
6694	Správa prístupu k editácii schémy	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6719	Zaznamenávanie novopridaných atribútov	Product Backlog Item	Bc. Jakub Janecek	Done	
6718	Full-text vyhľadávanie existujúcich atribútov	Product Backlog Item	Bc. Michal Sevcik	Committed	
6723	Dashboard certifikačnej schémy	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
6805	Refaktorovanie testov	Product Backlog Item	Bc. Martin Gulis	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6292	Obnova hesla	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6805	Refaktorovanie testov	Product Backlog Item	Bc. Martin Gulis	Done	
6694	Správa prístupu k editácii schémy	Product Backlog Item	Bc. Barbora Ungerova	Done	
6723	Dashboard certifikačnej schémy	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6718	Full-text vyhľadávanie existujúcich atribútov	Product Backlog Item	Bc. Michal Sevcik	Done	
6715	Full-text vyhľadávanie existujúcich metrík	Product Backlog Item	Bc. Michal Sevcik	Done	
6291	Zmena hesla používateľa	Product Backlog Item	Bc. Marek Vlha	Done	
6716	Zaznamenávanie novopridaných metrík	Product Backlog Item	Bc. Jakub Janecek	Done	
6719	Zaznamenávanie novopridaných atribútov	Product Backlog Item	Bc. Jakub Janecek	Done	
6804	Refaktorovanie dokumentácie	Product Backlog Item	Bc. Marek Vlha	Done	

Šprint 4 – Dinosaurus

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6697	Exportovanie zadaných údajov do RDF	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6721	Publikovanie opisu schémy	Product Backlog Item	Bc. Marek Vlha	Committed	
6772	Sledovanie výkonu servera	Product Backlog Item	Bc. Michal Sevcik	Committed	
6720	Schvaľovanie nových security atribútov	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
6717	Schvaľovanie nových metrík	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
7015	TP mentoring	Product Backlog Item	Bc. Jana	Comm	

			Tomcsanyiova	itted	
6699	Zobrazovanie porovnania medzi schémami	Product Backlog Item	Bc. Jakub Janecek	Comm itted	
6803	Refaktorovanie kódu aplikácie	Product Backlog Item	Bc. Martin Gulis	Comm itted	
6692	Rola reviewera opisu schémy	Product Backlog Item	Bc. Barbora Ungerova	Comm itted	
6770	Sledovanie chýb v backende	Product Backlog Item	Bc. Michal Sevcik	Comm itted	
7039	Secure server	Product Backlog Item	Bc. Michal Sevcik	Done	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6697	Exportovanie zadaných údajov do RDF	Product Backlog Item	Bc. Barbora Ungerova	Done	
6721	Publikovanie opisu schémy	Product Backlog Item	Bc. Marek Vlha	Done	
6772	Sledovanie výkonu servera	Product Backlog Item	Bc. Michal Sevcik	Done	
6720	Schvaľovanie nových security atribútov	Product Backlog Item	Bc. Denis Grotkovsky	Done	
6717	Schvaľovanie nových metrík	Product Backlog Item	Bc. Denis Grotkovsky	Done	
7015	TP mentoring	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
6699	Zobrazovanie porovnania medzi schémami	Product Backlog Item	Bc. Jakub Janecek	Done	
6803	Refaktorovanie kódu aplikácie	Product Backlog Item	Bc. Martin Gulis	Done	
6692	Rola reviewera opisu schémy	Product Backlog Item	Bc. Barbora Ungerova	Done	
6770	Sledovanie chýb v backende	Product Backlog Item	Bc. Michal Sevcik	Done	
7039	Secure server	Product Backlog Item	Bc. Michal Sevcik	Done	

Šprint 5 – Elektrický vláčik

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7236	Upravenie zobrazenia porovnania schém	Product Backlog Item	Bc. Jakub Janecek	Committed	
7234	Aktualizácia jquery	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
7233	Vytvorenie metodiky na manažment chýb	Product Backlog Item	Bc. Michal Sevcik	Committed	
7238	Finalizácia projektu na odovzdanie	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7231	Upravenie funkcionality priraďovania práv ku schéme	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7232	Upravenie RDF funkcionality	Product Backlog Item	Bc. Martin Gulis	Committed	
7230	Dokončenie šprintu 4	Product Backlog Item	Bc. Marek Vlha	Committed	
6771	Sledovanie chýb vo frontende	Product Backlog Item	Bc. Michal Sevcik	Done	
7229	Sledovanie výkonu servera	Product Backlog Item	Bc. Michal Sevcik	Done	
7239	Finalizácia dokumentácie na odovzdanie	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
7235	Zmenenie štruktúry kódu	Product Backlog Item	Bc. Marek Vlha	Committed	
7237	Upravenie schvaľovania nových entít	Product Backlog Item	Bc. Denis Grotkovsky	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7236	Upravenie zobrazenia porovnania schém	Product Backlog	Bc. Jakub Janecek	Done	

		Item				
7234	Aktualizácia jquery	Product Item	Backlog	Bc. Jana Tomcsanyiova	Done	
7233	Vytvorenie metodiky na manažment chýb	Product Item	Backlog	Bc. Michal Sevcik	Done	
7238	Finalizácia projektu na odovzdanie	Product Item	Backlog	Bc. Barbora Ungerova	Done	
7231	Upravenie funkcionality priraďovania práv ku schéme	Product Item	Backlog	Bc. Barbora Ungerova	Done	
7232	Upravenie RDF funkcionality	Product Item	Backlog	Bc. Martin Gulis	Done	
7230	Dokončenie šprintu 4	Product Item	Backlog	Bc. Marek Vlha	Done	
6771	Sledovanie chýb vo frontende	Product Item	Backlog	Bc. Michal Sevcik	Done	
7229	Sledovanie výkonu servera	Product Item	Backlog	Bc. Michal Sevcik	Done	
7239	Finalizácia dokumentácie na odovzdanie	Product Item	Backlog	Bc. Jana Tomcsanyiova	Done	
7235	Zmenenie štruktúry kódu	Product Item	Backlog	Bc. Marek Vlha	Committed	
7237	Upravenie schvaľovania nových entít	Product Item	Backlog	Bc. Denis Grotkovsky	Committed	

Šprint 6 – Furby

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7352	Zautomatizovanie prepisu 1/5 - vytváranie control objectives	Product Backlog Item	Bc. Jakub Janecek	Committed	
7413	Upravenie RDF exportu	Product Backlog Item	Bc. Barbora Ungerova	Committed	

7395	Neopakovanie sa atribútov v rámci jednej schémy	Product Backlog Item	Bc. Martin Gulis	Committed	
6728	Vytvorenie a zobrazenie prehľadu control questions	Product Backlog Item	Bc. Barbora Ungerova	Committed	
6701	Importovanie excelu s controls a control questions pre schému	Product Backlog Item	Bc. Michal Sevcik	Committed	
6730	Posudzovanie certifikačnej schémy	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
7358	Zobrazenie detailov porovnania control objectives	Product Backlog Item	Bc. Marek Vlha	Committed	
7597	Vytvorenie abstraktu a priebežnej správy	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
7415	Ukladanie hodnôt metrík (Min, Max, Between, Guaranteed) oddelene	Product Backlog Item	Bc. Michal Sevcik	Committed	
7353	Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7352	Zautomatizovanie prepisu 1/5 - vytváranie control objectives	Product Backlog Item	Bc. Jakub Janecek	Committed	
7413	Upravenie RDF exportu	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7395	Neopakovanie sa atribútov v rámci jednej schémy	Product Backlog Item	Bc. Martin Gulis	Done	
6728	Vytvorenie a zobrazenie prehľadu control questions	Product Backlog Item	Bc. Barbora Ungerova	Done	
6701	Importovanie excelu s controls a control questions pre schému	Product Backlog Item	Bc. Michal Sevcik	Done	
6730	Posudzovanie certifikačnej schémy	Product Backlog Item	Bc. Denis Grotkovsky	Done	

7358	Zobrazenie detailov porovnania control objectives	Product Backlog Item	Bc. Marek Vlha	Done	
7597	Vytvorenie abstraktu a priebežnej správy	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
7415	Ukladanie hodnôt metrík (Min, Max, Between, Guaranteed) oddelene	Product Backlog Item	Bc. Michal Sevcik	Done	
7353	Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	

Šprint 7 – Game Boy

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7353	Zautomatizovanie prepisu 2/5 - vyhľadávanie existujúcich atribútov a metrík	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
7355	Zautomatizovanie prepisu 4/5 - vytváranie nových metrík	Product Backlog Item	Bc. Marek Vlha	Committed	
7354	Zautomatizovanie prepisu 3/5 - vytváranie nových atribútov	Product Backlog Item	Bc. Michal Sevcik	Committed	
6729	Upravenie a odstraňovanie existujúcich schém, control-ov, control question-ov, control objective-ov	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
7413	Upravenie RDF exportu	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7352	Zautomatizovanie prepisu 1/5 - vytváranie control objectives	Product Backlog Item	Bc. Jakub Janecek	Committed	
7669	#27 Uncaught TypeError: Cannot set property 'onclick' of undefined	Bug	Bc. Barbora Ungerova	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
7353	Zautomatizovanie prepisu 2/5 - vyhľadávanie	Product	Bc. Jana Tomcsanyiova	Done	

	existujúcich atribútov a metrík	Backlog Item			
7355	Zautomatizovanie prepisu 4/5 - vytváranie nových metrík	Product Backlog Item	Bc. Marek Vlha	Done	
7354	Zautomatizovanie prepisu 3/5 - vytváranie nových atribútov	Product Backlog Item	Bc. Michal Sevcik	Done	
6729	Upravenie a odstraňovanie existujúcich schém, control-ov, control question-ov, control objective-ov	Product Backlog Item	Bc. Denis Grotkovsky	Done	
7413	Upravenie RDF exportu	Product Backlog Item	Bc. Barbora Ungerova	Done	
7352	Zautomatizovanie prepisu 1/5 - vytváranie control objectives	Product Backlog Item	Bc. Jakub Janecek	Done	
7669	#27 Uncaught TypeError: Cannot set property 'onclick' of undefined	Bug	Bc. Barbora Ungerova	Done	

Šprint 8 – Hrkálka

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6712	Naplnenie systému schémou CCM - Controls	Product Backlog Item	Bc. Marek Vlha	Committed	
7800	Upravenie fixtures	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7356	Zautomatizovanie prepisu 5/5 - plná automatizácia	Product Backlog Item	Bc. Michal Sevcik	Committed	
7798	Refaktorovanie databázy	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
7357	Zautomatizovanie prepisu control questions - plná automatizácia	Product Backlog Item	Bc. Martin Gulis	Committed	
7799	Dokončenie TP CUP článku	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
7924	#31 AttributeError: 'NoneType' object has no attribute 'security_attribute_name'	Bug	Bc. Jakub Janecek	Committed	

7929	#33 JSONDecodeError: Expecting value: line 1 column 1 (char 0)	Bug	Bc. Michal Sevcik	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6712	Naplnenie systému schémou CCM - Controls	Product Backlog Item	Bc. Marek Vlha	Committed	
7800	Upravenie fixtures	Product Backlog Item	Bc. Barbora Ungerova	Committed	
7356	Zautomatizovanie prepisu 5/5 - plná automatizácia	Product Backlog Item	Bc. Michal Sevcik	Done	
7798	Refaktorovanie databázy	Product Backlog Item	Bc. Denis Grotkovsky	Done	
7357	Zautomatizovanie prepisu control questions - plná automatizácia	Product Backlog Item	Bc. Martin Gulis	Done	
7799	Dokončenie TP CUP článku	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
7924	#31 AttributeError: 'NoneType' object has no attribute 'security_attribute_name'	Bug	Bc. Jakub Janecek	Done	
7929	#33 JSONDecodeError: Expecting value: line 1 column 1 (char 0)	Bug	Bc. Michal Sevcik	Done	

Šprint 9 – Igráček

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
8035	Upravenie generovania control objectivov	Product Backlog Item	Bc. Martin Gulis	Committed	
8045	Vytvorenie prvého návrhu posteru na TP CUP	Product Backlog Item	Bc. Jakub Janecek	Committed	
8031	Vytvorenie prezentácie tímu na TP CUP	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	

6706	Zaregistrovanie novej služby	Product Backlog Item	Bc. Jakub Janecek	Committed	
8046	Fixovanie bugov	Product Backlog Item	Bc. Barbora Ungerova	Committed	
8044	Pripravenie ISO 27k a testovanie importu	Product Backlog Item	Bc. Marek Vlha	Committed	
8033	Opravenie existujúcich control objectivov	Product Backlog Item	Bc. Martin Gulis	Committed	
8034	Prerobenie porovnávania schém	Product Backlog Item	Bc. Michal Sevcik	Committed	
8036	Vytvorenie staging branch	Product Backlog Item	Bc. Michal Sevcik	Committed	
7800	Upravenie fixtures	Product Backlog Item	Bc. Barbora Ungerova	Done	
7999	#40 ValueError: invalid literal for int() with base 10: ''	Bug	Bc. Barbora Ungerova	Committed	
7996	#36 IntegrityError: duplicate key value violates unique constraint "my_app_controlobjective_identifier_dd178442_uniq" DETAIL: Key (identifier)=(IAM-06.1) already exists.	Bug	Bc. Barbora Ungerova	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6706	Zaregistrovanie novej služby	Product Backlog Item	Bc. Jakub Janecek	Done	
8031	Vytvorenie prezentácie tímu na TP CUP	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
8033	Opravenie existujúcich control objectivov	Product Backlog Item	Bc. Martin Gulis	Done	
8034	Prerobenie porovnávania schém	Product Backlog Item	Bc. Michal Sevcik	Done	
8035	Upravenie generovania control objectivov	Product Backlog Item	Bc. Martin Gulis	Done	
8036	Vytvorenie staging branch	Product Backlog Item	Bc. Michal Sevcik	Done	
8044	Pripravenie ISO 27k a testovanie importu	Product Backlog Item	Bc. Marek Vlha	Done	

8045	Vytvorenie prvého návrhu posteru na TP CUP	Product Backlog Item	Bc. Jakub Janecek	Done	
8046	Fixovanie bugov	Product Backlog Item	Bc. Barbora Ungerova	Done	

Šprint 10 – Jojo

Stav úloh po prvom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6713	Naplnenie systému schémou ISO 27k (tabuľkové)	Product Backlog Item	Bc. Marek Vlha	Committed	
6995	Vytvorenie zmysluplných chybových stránok (4xx, 5xx)	Product Backlog Item	Bc. Denis Grotkovsky	Committed	
8032	Vytvorenie posteru TP CUP	Product Backlog Item	Bc. Denis Grotkovsky	Done	
8208	Pripravenie stránky na prezentovanie na TP CUP	Product Backlog Item	Bc. Martin Gulis	Committed	
8209	Pripravenie oviec na TP CUP	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
8227	Implementovanie procesu revieweovania schémy	Product Backlog Item	Bc. Barbora Ungerova	Committed	
8233	Vylepšenie graficko-používateľského rozhrania	Product Backlog Item	Bc. Jakub Janecek	Committed	
8234	Fixnutie konfliktu cookies	Product Backlog Item	Bc. Michal Sevcik	Done	
8236	Vylepšenie importu schémy	Product Backlog Item	Bc. Jakub Janecek	Committed	
8240	Vytvorenie tlačidla Skúsim šťastie	Product Backlog Item	Bc. Michal Sevcik	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
6713	Naplnenie systému schémou ISO 27k (tabuľkové)	Product Backlog Item	Bc. Marek Vlha	Done	

6995	Vytvorenie zmysluplných chybových stránok (4xx, 5xx)	Product Backlog Item		Bc. Denis Grotkovsky	Done	
8032	Vytvorenie posteru TP CUP	Product Backlog Item		Bc. Denis Grotkovsky	Done	
8208	Pripravenie stránky na prezentovanie na TP CUP	Product Backlog Item		Bc. Martin Gulis	Done	
8209	Pripravenie oviec na TP CUP	Product Backlog Item		Bc. Jana Tomcsanyiova	Done	
8227	Implementovanie procesu revieweovania schémy	Product Backlog Item		Bc. Barbora Ungerova	Done	
8233	Vylepšenie graficko-používateľského rozhrania	Product Backlog Item		Bc. Jakub Janecek	Done	
8234	Fixnutie konfliktu cookies	Product Backlog Item		Bc. Michal Sevcik	Done	
8236	Vylepšenie importu schémy	Product Backlog Item		Bc. Jakub Janecek	Done	
8240	Vytvorenie tlačidla Skúsim šťastie	Product Backlog Item		Bc. Michal Sevcik	Done	

Šprint 11 – Koniec

Stav úloh po prvom týždni:

ID	Title	Work Item Type		Assigned To	State	Tags
8411	Diseminovanie výsledkov	Product Backlog Item		Bc. Jakub Janecek	Committed	
8412	Finalizovanie projektu	Product Backlog Item		Bc. Denis Grotkovsky	Committed	
8421	Zinteligentnenie javascriptu	Product Backlog Item		Bc. Martin Gulis	Committed	
8422	Opravenie toho, že prepísanie 'en' na 'sk' zhodí aplikáciu	Product Backlog Item		Bc. Barbora Ungerova	Committed	
8423	Znovuposielanie chýb do rollbaru (aj keď sa zobrazí 500)	Product Backlog Item		Bc. Michal Sevcik	Committed	

8438	Zdokumentovaine finálneho projektu	Product Backlog Item	Bc. Jana Tomcsanyiova	Committed	
8445	Upravenie homepage a upravenie dashboardu schémy	Product Backlog Item	Bc. Marek Vlha	Committed	

Stav úloh po druhom týždni:

ID	Title	Work Item Type	Assigned To	State	Tags
8411	Diseminovanie výsledkov	Product Backlog Item	Bc. Jakub Janecek	Done	
8412	Finalizovanie projektu	Product Backlog Item	Bc. Denis Grotkovsky	Done	
8421	Zinteligentnenie javascriptu	Product Backlog Item	Bc. Martin Gulis	Done	
8422	Opravenie toho, že prepísanie 'en' na 'sk' zhodí aplikáciu	Product Backlog Item	Bc. Barbora Ungerova	Done	
8423	Znovuposielanie chýb do rollbaru (aj keď sa zobrazí 500)	Product Backlog Item	Bc. Michal Sevcik	Done	
8438	Zdokumentovaine finálneho projektu	Product Backlog Item	Bc. Jana Tomcsanyiova	Done	
8445	Upravenie homepage a upravenie dashboardu schémy	Product Backlog Item	Bc. Marek Vlha	Done	

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Rozpoznávanie cloudových služieb

[Ontosec]

Dokumentácia inžinierskeho diela

Vedúci tímu: Ing. Martin Labaj

Členovia tímu: Denis Grotkovský, Martin Gulis, Marek Vlha, Michal Ševčík, Barbora Ungerová, Jana Tomcsányiová, Jakub Janeček

Školský rok: 2017/2018

Obsah

Úvod

Tento dokument opisuje softvérový produkt vytvorený tímom O.F.C.S.A. (tím č. 12) na predmete Tímový projekt v akademickom roku 2017/2018. Témou, ktorú ako tím spracovávame, je rozpoznávanie cloudových služieb. Úlohou tímu je vytvoriť webovú aplikáciu, umožňujúcu formalizáciu, vytváranie a porovnávanie opisu cloudových služieb. Dokumentácia inžinierskeho diela má za cieľ obsiahnuť technickú špecifikáciu vytváranej aplikácie.

Globálne ciele

Hlavným cieľom projektu je vytvorenie webového portálu na manažment certifikačných schém. Certifikačná schéma opisuje požiadavky kladené na cloudové služby, ktoré sú formalizované pomocou ontológie. Našou motiváciou je umožniť cezhraničnú interoperabilitu a porovnávanie certifikačných schém, pričom ich formalizácia nielen uľahčí ich porovnávanie, ale aj umožní toto porovnávanie automatizovať. Výsledkom bude jednoduchšie využívanie cloudových služieb rôznymi používateľmi, ktorí budú aj z rôznych krajín, kde sa líši legislatíva týkajúca sa cloudových služieb.

Ciele pre zimný semester

Cieľom zimného semestra je vytvoriť webovú aplikáciu, ktorá umožní vytvoriť opis certifikačnej schémy a porovnať certifikačné schémy medzi sebou pre zistenie ich podobností. Opis schémy je definovaný ontológiou, ktorá definuje prvky, z ktorých sa skladá samotná schéma.

Prototyp aplikácie musí na konci semestra spĺňať nasledovnú funkcionalitu:

- správa používateľov,
- vytvorenie uceleného opisu certifikačnej schémy,
- manažment tvorby schémy,
- porovnanie certifikačných schém.

Každá z podmienok kladená na prototyp je rozpracovaná do väčšej hĺbky.

Správa používateľov

Aplikácia musí poskytovať používateľovi možnosť prihlásiť sa. Taktiež, ako administrátor musím mať možnosť vytvoriť nových používateľov a prideliť im práva. Potrebné je tiež umožniť používateľovi zmeniť heslo a obnoviť si heslo v prípade jeho zabudnutia.

Vytvorenie uceleného opisu certifikačnej schémy

Hlavným zameraním aplikácie je sformalizovanie opisu certifikačnej schémy pre účely jednoduchšieho porovnávania cloudových služieb, ktoré sú certifikované podľa daných opisov. Preto je nevyhnutné umožniť vytvoriť:

- certifikačnú schému,
- control pre certifikačnú schému,
- control objective pre control,
- bezpečnostný atribút pre control objective,
- metriku pre control objective.

Manažment tvorby schémy

Pri vytváraní schémy môže spolupracovať viacero používateľov. Potrebujeme evidovať, kto pracuje na ktorom opise schémy. Keď je schéma opísaná, potrebujeme určiť človeka, ktorý urobí kontrolu daného opisu. Až keď používateľ určený ako kontrolór schváli opis danej schémy, môže byť daná schéma publikovaná. Taktiež potrebujeme evidovať novovytvorené metriky a bezpečnostné atribúty. Schéma môže totiž byť publikovaná, len ak sú schválené všetky jej súčasti.

Porovnanie certifikačných schém

Využitie ontológie je kľúčové pre porovnanie opisu schém. Vieme vďaka nej porovnať zložky, z ktorých sa skladá opis schémy. To umožňuje vyhodnotiť podobnosť schém, a keďže schémami sú certifikované služby, vieme cez schémy porovnať aj tieto služby.

Ciele pre letný semester

Cieľom letného semestra je ďalej pracovať na vytvorenej webovej aplikácii. Funkcionalita určená na pridanie do aplikácie v letnom semestri zahŕňa automatický prepis (podporu prepisu) certifikačnej schémy z formy dokumentu excel do ontológie používanej aplikáciou. Zahŕňa taktiež podporu vykonávania auditov. Prototyp aplikácie musí na konci semestra spĺňať nasledovnú funkcionalitu:

- automatický prepis schémy,
- podpora vykonávania auditov.

Každá z podmienok kladená na prototyp je rozpracovaná do väčšej hĺbky.

Automatický prepis schémy

Aplikácia musí poskytovať používateľovi možnosť importovať certifikačnú schému vo formáte dokumentu excel, a jej následné spracovanie, ktorého výsledkom je vytvorená certifikačná schéma v našej aplikácii. K tejto schéme sú priradené aj control-y, ktoré sa nachádzali v danej schéme. Ďalej je nutné poskytnúť používateľovi pomoc pri transformácii opisov control-ov na control objective-y, bezpečnostné atribúty a metriky, aby bola schéma reprezentovaná nami vytvorenou ontológiou. Pri už spomínanej transformácii, je nutné využiť spracovanie prirodzeného jazyka (NLP – Natural Language Processing), aby sme vedeli identifikovať jednotlivé kľúčové prvky z opisov control-ov. Tie ďalej využívame na vytváranie nových prvkov ontológie. Tento proces nie je plne automatický, ale je pri ňom potrebný ľudský faktor kontroly.

Podpora vykonávania auditov

Cloudová služba zaregistrovaná v našej aplikácii si môže požiadať o vykonanie auditu podľa zvolenej certifikačnej schémy. V tom prípade auditor využie našu aplikáciu, aby si ukladal audit evidence k jednotlivým bodom auditu. Taktiež si môže evidovať progres v audite.

Zhodnotenie dosiahnutia cieľov

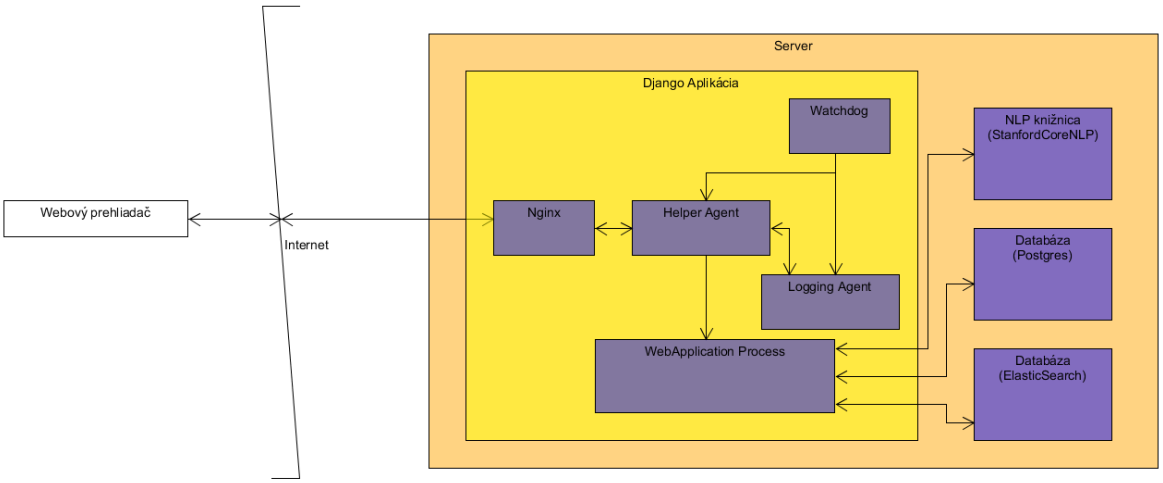
Dôsledkom náročnosti spracovania NLP, ktorá prekročila očakávané hodnoty, a z toho vyplývajúcich požiadaviek product owner-ov, sme nakoniec nezačali pracovať na dosiahnutí cieľa **Podpora vykonávania auditov**.

Celkový pohľad na systém

Táto časť poskytuje opis architektúry a dátového modelu vytváraného systému.

Architektúra

Systém je založený na štýle klient-server. Rolu klienta tu zastáva webový prehliadač, čiže veľmi tenký klient. Server zabezpečuje biznis logiku a údaje pre systém. Využívajú sa dva typy databázy – PostgreSQL pre bežnú prácu s dátami a ElasticSearch pre fulltextové vyhľadávanie v systéme. NLP knižnicu StanfordCoreNLP využívame pri spracovávaní prirodzeného jazyka používaného na opis požiadaviek v certifikačných schémach. Samotná aplikácia je servovaná pomocou aplikačného servera.

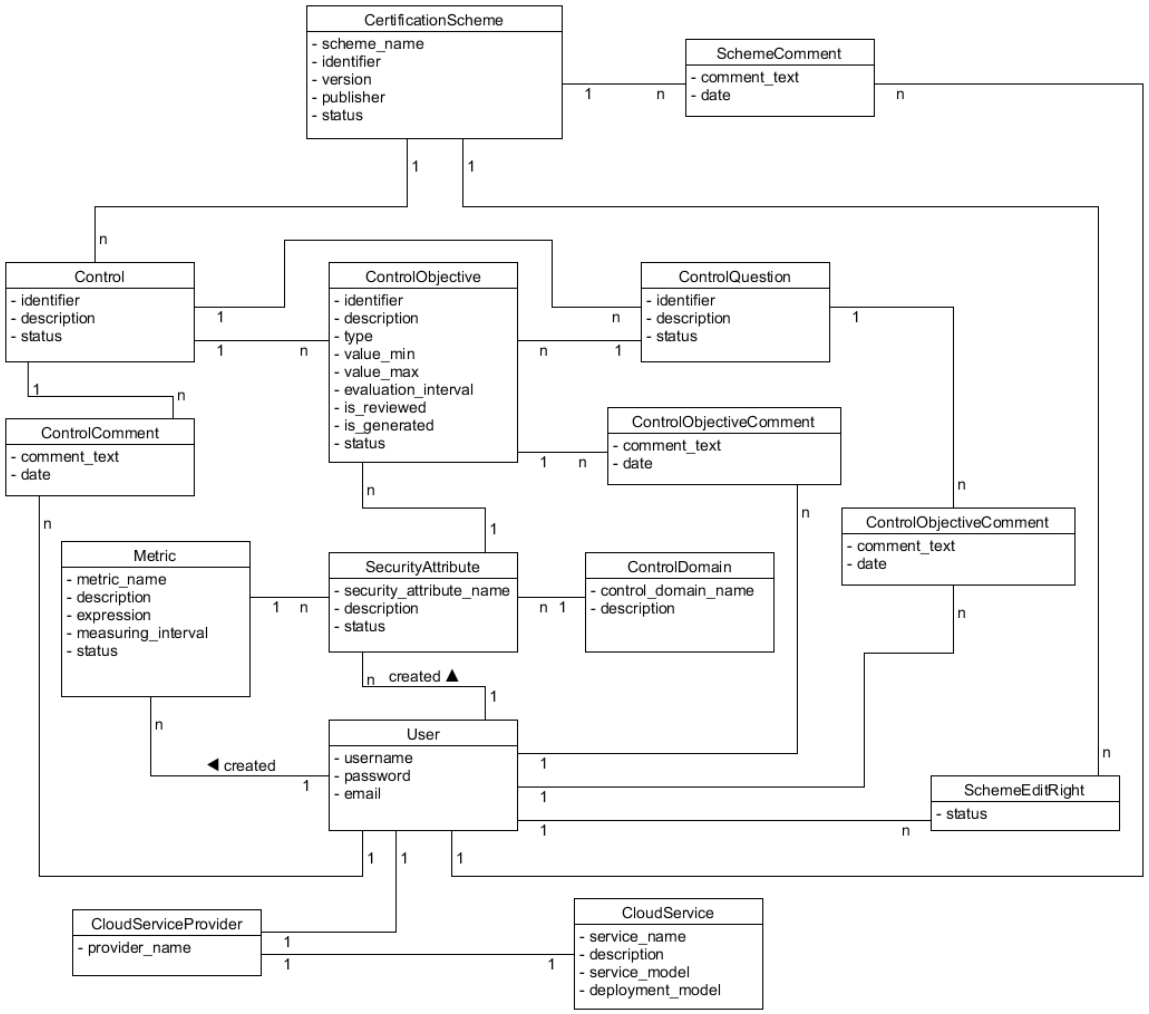


Obrázok 1: Architektúra systému.

Dátový model

Diagram dátového modelu reprezentuje spôsob uloženia údajov v databáze. *CertificationScheme* je hlavná entita. Zastáva úlohu celkového opisu certifikačnej schémy pre cloudovú službu. Skladá sa z *Control*-ov, ktoré hovoria o jednotlivých oblastiach služby, v ktorých majú byť splnené jednotlivé *ControlObjective*-y. Tie hovoria o konkrétnych bodoch, ktoré musí služba spĺňať, aby mohla byť certifikovaná danou schémou. Ku *Control-om* môžu taktiež prislúchať *ControlQuestion-s*, ktoré sa taktiež skladajú z *ControlObjective-ov*. Tieto body (control objective-y) sú definované *SecurityAttribute*-om. *SecurityAttribute* patrí do vybranej *ControlDomain* a je meraný priradenou *Metric*-ou. *SchemeEditRight* slúži na manažovanie prístupu používateľov k opisu schém. *User* reprezentuje používateľa aplikácie a využíva sa pri zaznamenávaní autora niektorých častí ontológie a pri komentovaní v procese schvaľovania opisu schémy.

User sa taktiež používa na reprezentáciu používateľa, ktorý sa chce zaregistrovať ako *CloudServiceProvider* , teda poskytovateľ cloudovej služby. Na neho môže byť naviazaných viacero *CloudService-s*.



Obrázok 2: Dátový model.

Moduly

Hlavné rozdelenie pri opise modulov je na server a jeho nastavenie, a samotnú webovú aplikáciu, ktorá je opísaná možnosťami použitia.

Správa používateľov

Na správu používateľov sme sa rozhodli využiť funkcionalitu, ktorú Django ponúka predimplementovanú. Využívame ju na prihlásenie používateľa, vytváranie ďalších používateľských kont, zmenu hesla a obnovu hesla. Pri týchto úlohách sme danú funkcionalitu buď iba využili, alebo upravili tak, aby viac vyhovovala našim potrebám.

Prihlásenie používateľa

Analýza

Funkcionalita aplikácie je dostupná len pre prihláseného používateľa. Jeho účet je následne v aplikácii využívaný na spravovanie prístupu a ukladanie autorov novovytvorených záznamov. Preto je potrebné poskytnúť používateľovi možnosť prihlásiť sa.

Návrh

Pre používateľa potrebujeme vedieť jeho prihlasovacie meno, heslo, email.

Implementácia

Rozhodli sme sa využiť funkcionalitu Django framework-u, ktorý poskytuje možnosť využiť predimplemenovanú funkcionalitu na prihlásenie a len ju zapracovať do vytváranej aplikácie.

Testovanie

Testovanie prihlásenia používateľa prebieha prostredníctom jednotkových aj integračných testov. Jednotkovými testami sa zameriavame na testovanie vyžiadania prihlasovacích údajov pri prístupe na podstránky. Integračnými testami prebieha proces prihlásenia aj odhlásenia používateľa. Testy zahŕňajú správny prípad použitia pri prihlásení, ale aj nesprávne scenáre pri zadaní nesprávnych údajov alebo nevyplnení polí. Okrem testovania cookies, či došlo k uloženiu spojenia, overujeme aj presmerovanie adresy po zadaní správnych údajov na korektnú podstránku alebo zotrvanie na podstránke po zadaní chybných údajov. Tiež testujeme korektnosť odkazov, ktorými sa dá dostať na podstránku s prihlásením alebo odhlásením.

Vytváranie ďalších používateľských kont

Analýza

Systém bude používaný viacerými používateľmi naraz, častokrát s obmedzenými právami, a preto je nutné vytvárať nových používateľov.

Návrh
Je nutné vytvoriť prostredie, ktoré umožní používateľovi systému vytvoriť nového používateľa, vyplniť mu údaje a prideliť mu potrebné práva do systému.

Implementácia
Potrebná funkcionalita je poskytnutá vo framework-u Django. Na stránke pribudlo prepojenie na stránku admin, ktorá umožní prihláseným používateľom s právami pridať nových používateľov. Taktiež sa môže používateľ zaregistrovať cez formulár.

Testovanie
Značná časť funkcionalít je závislá od Django framework-u. Presný obsah využívaných funkcií preto nemôže byť nami dôkladne testovaný jednotkovými testami. Integračnými testami overujeme proces vytvárania nových používateľských kont.

Zmena hesla používateľa
Analýza
Používateľ je vytváraný iným používateľom, a preto je nutné, aby si mohol svoj účet sám zabezpečiť. Na toto bude slúžiť zmena hesla používateľa.

Návrh
Vytvoriť formulár pre používateľa na zmenu hesla (zadanie starého hesla, nového hesla a potvrdenie nového hesla).

Implementácia
Potrebná funkcionalita je poskytnutá vo framework-u Django. Na stránke pribudlo prepojenie na stránku admin, ktorá umožní prihláseným používateľom zmeniť heslo pomocou dostupného formuláru.

Testovanie
Zmena hesla používateľa je závislá od Django framework-u, preto nemôže byť testovaný dôkladne jednotkovými testami. Proces zmeny hesla je overovaný integračnými testami.

Obnovenie hesla
Analýza
Môže sa stať, že používateľ zabudne heslo od svojho používateľského konta. Aby sa požiadavka na vytvorenie nového hesla neposielala stále adminovi, je potrebné vytvoriť štandardnú obnovu hesla pomocou odoslania odkazu na obnovu hesla na e-mailovú adresu používateľa.

Návrh
Vytvorenie možnosti obnovenia hesla pre používateľa. Vytvorenie formuláru, kde môže používateľ zadať svoje používateľské meno alebo svoj e-mail a kde mu je následne odoslaná

správa obsahujúca odkaz na vytvorenie nového hesla. Následne vytvorenie formulára, kde používateľ môže zadať nové heslo.

Implementácia

Na stránke pre prihlásenie bol pridaný odkaz na obnovu hesla. Na frontend-e bol vytvorený formulár, kde používateľ zadá meno alebo e-mail. Následne sa na backend-e tento vstup spracuje, zistí sa, či používateľ zadal svoje meno alebo e-mail, prejdú sa všetky používateľské kontá a zistí sa, či daný používateľ existuje v systéme. Ak nie, systém na to používateľa upozorní. Ak áno, systém vytvorí odkaz na obnovu hesla, ktorý pošle s automaticky vygenerovanou správou na e-mail používateľa. Po kliknutí na odkaz sa vytvorí formulár pre vytvorenie nového hesla. Po zadaní nového hesla systém heslo zmení a používateľ sa následne môže prihlasovať s novým heslom.

Testovanie

Testovanie tejto časti aplikácie zahŕňa jednotkové aj integračné testy. Testujú sa aj chybové správy v prípady zadania neexistujúceho e-maily.

Vytváranie a správa opisu certifikačnej schémy ontológiou

Opis certifikačnej schémy ontológiou vzniká postupne krokmi, pre ktoré sme vytvorili formuláre a zobrazenia, pomocou ktorých používateľ môže zapísať danú certifikačnú schému. Vďaka využitiu ontológie bude možné tieto opisy porovnávať. Umožňujeme vytvorenie novej schémy, zobrazenie prehľadu schém, detailu schémy, vytvorenie control-u ku schému, zobrazenie jeho detailu a pridanie control objective-u k nemu. Umožňujeme taktiež používateľovi, ktorý je správca alebo vlastník schémy, aby upravil prístupové práva k schéme ostatným používateľom.

Vytvorenie novej certifikačnej schémy

Analýza

Certifikačná schéma je základom ontológie pre jej opis. Ďalej sa skladá z control-ov. Certifikačná schéma je pri vytváraní identifikovaná nasledovnými atribútmi:

- meno schémy,
- vydavateľ,
- identifikátor,
- verzia.

Používateľ musí mať možnosť vytvoriť novú schému.

Návrh

Nebol potrebný komplexný návrh, keďže sa jedná o jednoduchú entitu.

Implementácia

Vytvorili sme formulár, v ktorom používateľ vyplní atribúty uvedené v analýze. Keď zadá všetky údaje a potvrdí vytvorenie, server spracuje požiadavku a ak nie je so zadanými údajmi

žiadny problém, uloží záznam do databázy. Používateľ je presmerovaný na prehľad certifikačných schém.

Testovanie

Testovanie tejto časti prebieha jednotkovými aj integračnými testami. Zobrazenie podstránky, využitie správneho template a vyžiadanie prihlásenia používateľa pri vstupe na podstránku testujeme prostredníctvom jednotkových testov. Časť modelu pre schému je pokrytá tiež jednotkovými testami. Integračné testy overujú dodržiavanie neprázdnych polí pri vytváraní schémy a jej vytvorenie pri správnom zadaní vstupných údajov. Okrem samotného pridania záznamu do databázy testujeme aj presmerovanie z tejto podstránky na prehľad certifikačných schém.

Prehľad certifikačných schém

Analýza

V aplikácii potrebuje mať používateľ prehľad o tom, aké schémy sa už v systéme nachádzajú.

Návrh

Tento prehľad musí zobrazovať všetky dôležité údaje pre každú schému, aby sa už z tohto prehľadu používateľ dozvedel čo najviac informácií o schéme.

Implementácia

Prehľad je implementovaný pomocou tabuľky, ktorá zobrazuje všetky údaje zadávané pri vytváraní novej certifikačnej schémy a taktiež počet control-ov, ktoré danú schému opisujú. Z tohto prehľadu sa môže používateľ prekliknúť na pridanie novej schémy a taktiež môže zobraziť jej detail.

Testovanie

Zobrazenie podstránky, využitie správneho template a vyžiadanie prihlásenia používateľa pri vstupe na podstránku testujeme prostredníctvom jednotkových testov. Integračné testovanie sa zameriava na korektné zobrazenie záznamov v tabuľke a správne presmerovania do ostatných častí aplikácie pri interakcii s podstránkou.

Detail certifikačnej schémy

Analýza

Pri detaile schémy používateľ potrebuje vidieť čo najviac informácií o schéme, čím sa nemyslia len jej atribúty. Potrebuje vidieť štatistiky o vybranej schéme a o stavebných prvkoch, ktoré ju definujú.

Návrh

Detail schémy by mal uvádzať jej meno, identifikátor a vydavateľa, spolu so zoznamom control-ov, ktoré ju opisujú. Taktiež by mal uvádzať údaje o tom, či je schéma schválená, publikovaná, aký je stav jej control-ov a control objective-ov.

Implementácia

Detail schémy je implementovaný pomocou niekoľkých textov a dvoch tabuliek. Jedna tabuľka je dashboard schémy, ktorý obsahuje štatistické údaje o schéme. Druhou tabuľkou je zoznam control-ov. Používateľ sa vie prekliknutím dostať k detailu control-u, alebo k pridaniu nového control-u.

Testovanie

Zobrazenie podstránky, využitie správneho template a vyžiadanie prihlásenia pri vstupe na podstránku používateľa testujeme prostredníctvom jednotkových testov. Integračné testovanie sa zameriava na korektné zobrazenie záznamov v tabuľkách a správne presmerovania do ostatných častí aplikácie pri interakcii s podstránkou. Zároveň testujeme aj prístupové práva k detailu konkrétnej schémy.

Vytvorenie nového control-u

Analýza

Control je prvok certifikačnej schémy, ktorý definuje, čo má overiť audit vykonávaný na základe certifikačnej schémy, ktorá obsahuje spomínaný control. Control je priradený k jednej schéme a je definovaný nasledovnými atribútmi:

- identifikátor,
- popis,
- text.

Používateľ musí mať možnosť vytvoriť nový control.

Návrh

Nebol potrebný komplexný návrh, keďže sa jedná o jednoduchú entitu.

Implementácia

Vytvorili sme formulár, v ktorom používateľ vyplní atribúty uvedené v analýze. Keď zadá všetky údaje a potvrdí vytvorenie, server spracuje požiadavku a ak nie je so zadanými údajmi žiadny problém, uloží záznam do databázy. Používateľ je presmerovaný na detail certifikačnej schémy, ku ktorej control vytvoril.

Testovanie

Zobrazenie podstránky, využitie správneho template a vyžiadanie prihlásenia používateľa pri vstupe na podstránku testujeme prostredníctvom jednotkových testov. Control v časti model je pokrytý jednotkovými testami. Integračné testy overujú dodržiavanie neprázdnych polí pri vytváraní a pridanie záznamu do databázy pri správnom zadaní vstupných údajov. Testujeme aj všetky odkazy a presmerovania z tejto podstránky do ostatných častí aplikácie.

Detail control-u

Analýza

Detail control-u je rovnaký prípad ako detail schémy, akurát tu už nepotrebujeme štatistické údaje, a namiesto zoznamu control-ov chceme vidieť zoznam control objective-ov.

Návrh

Detail control-u by mal uvádzať jeho meno, ku ktorej schéme patrí, jeho opis a taktiež zoznam control objective-ov, ktoré slúžia na kontrolu pri audite pre daný control.

Implementácia

Detail control-u je implementovaný pomocou niekoľkých textov a tabuľky. Tabuľka obsahuje zoznam control objective-ov. Používateľsa vie dostať k pridaniu nového control objective-u.

Testovanie

Zobrazenie podstránky, využitie správneho template a vyžiadanie prihlásenia používateľa pri vstupe na podstránku testujeme prostredníctvom jednotkových testov. Integračné testovanie sa zameriava na korektné zobrazenie záznamov v tabuľke a správne presmerovania do ostatných častí aplikácie pri interakcii s podstránkou.

Vytvorenie nového control objective-u

Analýza

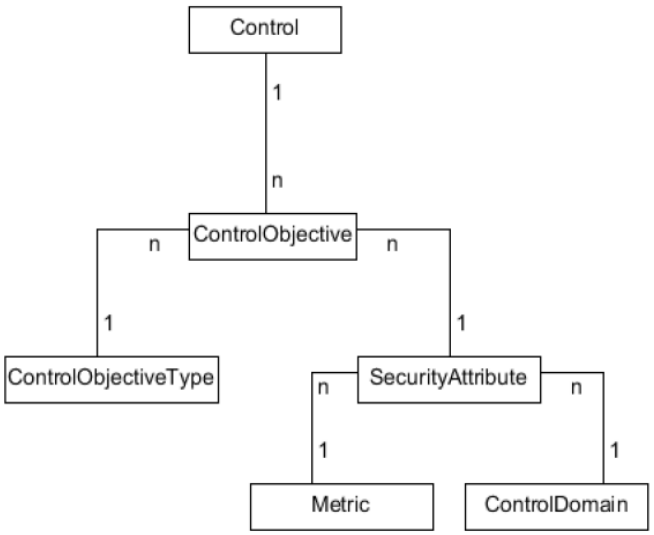
Pre jednotlivé schémy môže existovať niekoľko control-ov. Tieto control-y obsahujú control objective-y. Jeden control objective je opísaný pomocou niekoľkých atribútov. Medzi tieto atribúty patria napríklad:

- unikátny identifikátor,
- bezpečnostný atribút,
- metrika bezpečnostného atribútu.

Náš systém musí dať možnosť používateľovi jednoducho a intuitívne pridať takýto control objective k danému control-u.

Návrh

Aby sme zvládli správne implementovať takúto funkcionalitu, musíme si lepšie definovať štruktúru jedného control objective-u. Na obrázku je znázornený dátový model pre control objective.



Obrázok 1: Schéma pre Control Objective.

Implementácia

Tento problém sme implementovali pomocou formuláru, ktorý je validovaný a následne uložený do databázy. Bol vytvorený formulár, do ktorého sa zadávajú nasledovné údaje:

- identifikátor – musí byť unikátny pre control objective,
- opis – textový opis control objective-u.

Následne je možné vybrať, či chceme vytvoriť nový bezpečnostný atribút alebo nájsť už existujúci. Bezpečnostný atribút je definovaný pomocou:

- mena atribútu – unikátne,
- opis – textový opis bezpečnostného atribútu,
- metrika – metrika bezpečnostného atribútu.

Metrika môže byť tak isto ako bezpečnostný atribút vytvorená alebo použitá už existujúca. Metrika sa skladá z nasledujúcich atribútov:

- názov metriky,
- opis – textový opis metriky,
- vyjadrenie metriky,
- interval metriky.

Pri vytváraní metrík a atribútov sa k novovytvoreným entitám priradí aktuálne prihlásený používateľ ako autor, a tieto entity sa označia statusom ako nové, čo značí, že ich je ešte potrebné schváliť.

Všetky potrebné dáta sú uložené v databáze PostgreSQL. Modely sa ukladajú pomocou ORM mapovača, ktorý priamo poskytuje Django framework.

Pri vytváraní má používateľ možnosť vyhľadávať v existujúcich metrikách. Toto vyhľadávanie prebieha pomocou ElasticSearch. V ElasticSearch sú naindexované všetky metriky, ktoré v databáze existujú. Rovnako poskytujeme funkciu na naindexovanie už existujúcich metrík. V ElasticSearch sú namapované jednotlivé metriky z databázy. Obsahujú názov a ID z databázy.

Z frontend-u sa požiadavky posielajú na server pomocou AJAX. Vyvolá sa javascript funkcia, ktorá odošle požiadavku. Pomocou JSON sa presúvajú dáta medzi frontend-om a backend-om.

Testovanie

Testovanie vytvárania nového control objective-u zahŕňa jednotkové aj integračné testy. Oba typy testov sa zameriavajú na možné scenáre vytvárania objektov.

Nastavenie prístupových práv pre schému

Analýza

Ako vlastník opisu schémy (teda osoba, ktorá ju vytvorila) alebo správca, chceme mať možnosť nastaviť používateľom, či môžu opis schémy upravovať alebo nie. Preto potrebujeme mať možnosť nastaviť práva k schémam pre jednotlivých používateľov.

Návrh

Potrebujeme evidovať práva, pričom každý používateľ môže mať práva k niekoľkým opisom schém, a každý opis schémy môže mať možnosť evidovať niekoľko používateľov. Taktiež potrebujeme evidovať, kto je vlastníkom opisu. Aby sme sa pripravili na historizáciu týchto práv, rozhodli sme sa vzťah prístupu k schéme reprezentovať entitou *SchemeRights*. Právo pre vlastníka sa vytvorí vždy pri vytváraní novej schémy.

Implementácia

Nastavenie prístupových práv pre opis schémy je dostupné zo stránky detailu schémy, ale iba v prípade, ak je prihlásený používateľ vlastník schémy alebo správca. Následne pomocou zaškrtávacích políčok vyberie používateľov, ktorí majú mať právo upravovať opis schémy a uloží svoje rozhodnutie.

Testovanie

Testovanie tejto podstránky prebieha pomocou jednoduchých jednotkových aj integračných testov. Vzhľadom k tomu, že táto funkcionalita je pomerne jednoduchá, testy nie sú príliš rozsiahle.

Schvaľovanie entít

Analýza

Správca potrebuje schváliť, zamietnuť alebo upraviť novovzniknuté metriky a bezpečnostné atribúty. Potrebuje vidieť detaily týchto entít, aké iné atribúty sú na ne naviazané a podobné entity, ktoré už existujú.

Návrh

Správca si bude môcť vybrať, či chce schvaľovať metriky alebo bezpečnostné atribúty. Následne vyberie zo zoznamu metrík alebo bezpečnostných atribútov požadovanú entitu na schvaľovanie. Túto entitu bude môcť schváliť, zamietnuť alebo zmeniť. Tiež mu bude poskytnutý zoznam podobných entít podľa mena a opisu.

Implementácia

Správca si v hlavnom panely vyberie, či chce pracovať s metrikami alebo bezpečnostnými atribútmi. Následne je vytvorený jednoduchý formulár, kde sa mu zobrazia všetky nové entity. Sú zobrazené v tabuľke, kde administrátor vidí ich atribúty. Kliknutím na entitu ju vyberie. Je vytvorený ďalší formulár, kde správca schvaľuje, zamieta alebo mení atribúty entity. Tam je zobrazená vybraná entita so všetkými svojimi atribútmi. Ďalej je používateľovi poskytnutá možnosť schválenia, zamietnutia alebo úpravy.

- Schválenie − správca je po schválení entity presmerovaný na stránku výberu entít. Entita je označená ako schválená a nie je obsiahnutá v zozname entít na výber.
- Zamietnutie − tak ako pri schválení, avšak entita je označená ako zamietnutá.
- Úprava − správcovi sa zobrazí formulár na zmenu atribútov, kde tieto atribúty môže zmeniť zápisom do príslušného poľa. Nemusí zadávať atribúty do všetkých polí, len do tých, ktoré chce zmeniť. Následne potvrdí zmenu tlačidlom. Atribúty sa

nasledovne zmenia a správca môže pokračovať so schválením alebo zamietnutím aktuálnej entity.

Správcovi je tiež poskytnutý zoznam podobných entít. Entity sa na podobnosť porovnávajú pomocou ElasticSearch. Porovnáva sa ich názov a opis. Porovnávajú sa všetky entity, nie len tie nové.

Testovanie

Testovanie daného procesu sa vykonáva jednotkovými aj integračnými testami. Testuje sa formulár a view funkcia spolu so správnym počtom zobrazených entít v html.

Vytvorenie roly recenzenta

Analýza

Po schválení schémy je potrebné danú schému skontrolovať, či obsahuje všetky potrebné údaje. Toto robí recenzent, ktorého si môže určiť vlastník schémy alebo správca aplikácie. Recenzent nesmie mať právo schému upravovať.

Návrh

Potrebujeme evidovať roly, pričom každý používateľ môže mať rolu recenzenta k niekoľkým opisom schém, a každý opis schémy môže mať možnosť evidovať niekoľko recenzentov. Taktiež potrebujeme evidovať, kto je vlastníkom opisu. Rola recenzenta sa nachádza v rovnakej entite ako aj nastavovanie práv pre používateľov, teda v entite *SchemeRights*. Recenzent nesmie mať právo upravovať certifikačnú schému, teda právo na úpravu a rola recenzenta musia mať vždy opačnú hodnotu.

Implementácia

Nastavenie roly recenzenta pre opis schémy je dostupné zo stránky detailu schémy, ale iba v prípade, ak je prihlásený používateľ vlastník schémy alebo správca. Následne pomocou zaškrtávacích políčok vyberie používateľov, ktorí majú mať právo recenzovať opis schémy a uloží svoje rozhodnutie.

Testovanie

Testovanie tejto podstránky prebieha pomocou jednoduchých jednotkových aj integračných testov. Vzhľadom k tomu, že táto funkcionalita je pomerne jednoduchá, testy nie sú príliš rozsiahle.

Publikovanie certifikačnej schémy

Analýza

Aby bolo možné certifikačné schémy efektívne porovnať, je nutné, aby prešli jej entity podrobnou kontrolou a následným schválením. Takéto schválenie je nutné umožniť vlastníkovi certifikačnej schémy.

Návrh
Publikovanie je navrhnuté tak, aby sa zobrazilo v detaile certifikačnej schémy, kde je možné skontrolovať všetky potrebné entity.

Implementácia
Publikovanie certifikačnej schémy je implementované ako jedno tlačidlo, ktoré sa zobrazí iba vlastníkovi certifikačnej schémy v prípade, že je daná certifikačná schéma v stave schválená.

Testovanie
Testovanie prebehlo jednotkovými testami a integračnými testami.

ElasticSearch
Modul využívaný pre full-text vyhľadávanie a odporúčanie existujúcich metrík a bezpečnostných atribútov.

Analýza
Pri vytváraní nového control objective-u je potrebné vyhľadávať, napríklad v existujúcich metrikách alebo v existujúcich bezpečnostných atribútoch. ElasticSearch je tiež možné využiť pri vyhľadávaní podobných entít v databáze, aby sa predišlo duplicite.

Návrh
Treba napojiť ElasticSearch na server. Treba vytvoriť indexovanie potrebných entít z PostgreSQL do ElasticSearch. Takisto treba mapovať novovytvorené entity do ElasticSearch. A keď sa entita maže, musí sa zmazať aj z ElasticSearch.

Implementácia
ElasticSearch beží na serveri tímu. Každá entita, ktorá musí byť mapovaná do ElasticSearch, musí implementovať funkciu indexing, ktorá operuje s reprezentáciou objektu v ElasticSearch. Existujú tiež metódy bulk indexing, ktoré namapujú všetky doteraz existujúce entity z PostgreSQL do ElasticSearch. Na indexovanie a mazanie entít sa používajú signály, kde sa po uložení do databázy zavolá indexing nad novovytvorenou entitou. Ak sa entita zmaže, spustí sa na zmazanej inštancii remove indexing a tým sa odstráni z ElasticSearch.

Testovanie
ElasticSearch má vytvorený vlastný index pre testy, ktorý sa automaticky aktualizuje počas testovania.

RDF Export
Modul sa využíva na exportovanie certifikačných schém, control-ov, control objective-ov, jeho bezpečnostných atribútov a metrík do formátu RDF.

Analýza

V aplikácií sú jednotlivé entity prepojené. Toto prepojenie je potrebné vyjadriť v štruktúrovanej podobe, v ktorej sa zachovajú vzťahy medzi entitami. Na toto vyjadrenie bol ako najvhodnejší formát vybraný formát RDF.

Návrh

Je potrebné získať entity z databázy, ako aj poskytnúť používateľom možnosť exportovať len vybrané entity. Získané entity je potrebné upraviť do vybraného formátu RDF a následne umožniť používateľovi stiahnuť súbor s danými entitami v RDF formáte.

Implementácia

V aplikácii je používateľovi umožnené generovať do RDF certifikačné schémy, control-y, control objective-y, bezpečnostné atribúty a metriky, spolu so všetkými údajmi z databázy každej entity. Uvedené údaje sa generujú do formátu RDF pomocou funkcie z knižnice rdflib. Ako základné uri pre entity sme použili uri: http://fiit.stuba.sk/eusec#, za ktorým sa nachádza pomenovanie danej entity a následne obsah entity (príklad: http://fiit.stuba.sk/eusec#certificationScheme/Profi). Vybrali sme RDF formát ntriples, ktorého zápis je napríklad: <http://fiit.stuba.sk/eusec#certificationScheme/Profi> <http://fiit.stuba.sk/eusec#/hasControl> <http://fiit.stuba.sk/eusec#control/data> . Vygenerované RDF trojice sa uložia do súboru a poskytnú požívateľovi na stiahnutie.

Testovanie

Testovanie základnej funkcionality súvisiacej s korektným načítaním stránky sú implementované prostredníctvom jednotkových testov. Následne sú vytvorené aj testy nad dátami pripomínajúce reálne dáta na verifikáciu generovania korektných údajov.

Porovnanie certifikačných schém

Porovnanie certifikačných schém sa vykonáva na základe bezpečnostných atribútov a control objective-ov, ktoré opisujú schému. Control objective je prvok, ktorý sa už skladá z bezpečnostných atribútov a prvkov, ktoré sú spoločné medzi opismi schém. Teda control objective je definovaný bezpečnostným atribútom a metrikou. V inej schéme môže byť iný control objective, ktorý bude definovaný rovnakým bezpečnostným atribútom a metrikou. To nám umožňuje medzi sebou dané control objective-y porovnať, a teda porovnať aj samotné schémy. V schémach sa najskôr nájdu spoločné bezpečnostné atribúty a pre tieto bezpečnostné atribúty sa porovnajú control objective-y, v ktorých sa nachádzajú.

Výber certifikačných schém na porovnanie

Analýza

Používateľ si chce porovnať dve schémy medzi sebou, aby vedel, aká je ich podobnosť, a čo spĺňa z druhej, ak spĺňa prvú.

Návrh

Prihlásený používateľ musí mať možnosť vybrať si schémy, ktoré chce medzi sebou porovnať.

Implementácia

Proces je implementovaný pomocou formuláru, kde si používateľ v dvoch rozbaľovacích zoznamoch zvolí dve schémy, ktoré sa majú medzi sebou porovnať. Svoju voľbu potvrdí tlačidlom.

Testovanie

Testovanie daného procesu sa vykonáva jednotkovými testami, ktoré otestujú formulár definovaný form-om a view funkciu, ktorá obsahuje logiku za formulárom. Existujú taktiež integračné, ktoré overujú možnosť vybratia dvoch schém.

Porovnanie certifikačných schém

Analýza

Používateľ potrebuje vedieť, aké sú vzťahy medzi control objective-mi porovnávaných schém. Chce taktiež vedieť, ako je celkovo jedna schéma podobná druhej.

Návrh

Prihlásenému používateľovi sa po zvolení dvoch certifikačných schém na porovnanie zobrazí toto porovnanie. Následne má možnosť zobraziť si podrobnosti jednotlivých porovnaní medzi bezpečnostnými atribútmi, pod ktorými sú porovnania medzi control objective-mi.

Implementácia

Porovnanie schém pozostáva z textového opisu, ktorý hovorí o tom, ako sa schémy mapujú jedna na druhú. Toto mapovanie je vyjadrené v pomere control objectove-ov. Následne sú zobrazené tri tabuľky. Prvé dve tabuľky obsahujú bezpečnostné atribúty a k ním

prislúchajúce control objective-y, ktoré sa nachádzajú buď iba v jednej alebo iba v druhej schéme. Tretia tabuľka obsahuje bezpečnostné atribúty a ich control objective-y z oboch schém, ktoré sa na seba namapovali. Namapujú sa na seba tie control objective-y, ktoré sú definované rovnakým bezpečnostným atribútom. U nich sa následne porovná typ a hodnota pre tento typ. Typ sa porovná na úrovni reťazcového porovnania. Rozdiely sa vyfarbia červenou farbou. Spoločné časti sa vyfarbia zelenou.

Testovanie

V prvej verzii tohto procesu bolo testovanie zastúpené jednotkovými testami. Avšak kvôli zmenám sú momentálne testy vo fáze úpravy.

Detail control objective-u porovnávaných certifikačných schém

Analýza

Pri porovnávaní certifikačných schém a prípadných nezhodách v control objective-och je žiadané zobrazenie všetkých informácií o control objective pre používateľa.

Návrh

V obrazovke porovnávaných certifikačných schém sa nachádza tlačidlo na zobrazenie detailu control objective-ov.

Implementácia

Detail control objective-u je implementovaný ako modálne okno, ktoré sa zobrazí používateľovi po stlačení tlačidla s informačným symbolom, ktoré sa nachádza pri každom control objective. V modálnom okne sú zobrazené všetky informácie v tabuľke. Informácie o control objective, ktoré obsahujú iba jeden control objective, nebudú mať vyplnené informácie pre druhý control objective.

Testovanie

Testovanie je zabezpečené integračnými testami.

Server

Nastavenia servera

Na server sa pripája pomocou SSH, pričom máme prístup aj k vzdialenému serveru cez virtuálnu sieť. Na overenie totožnosti prihlasovaného používateľa sa používajú RSA kľúče.

Web server beží na doméne team12-17.studenti.fiit.stuba.sk. Webová prezentácia beží na klasickom http porte :80.

Aplikačný server beží pomocou Passenger-a. Passenger sa do Nginx-u pridal pomocou špeciálnej konfigurácie Nginx-u, ktorá sa nachádza na bežnom mieste v myapp.conf.

```
server {
   passenger_python /usr/bin/python3.6;
   listen 80;
   server_name team12-17.studenti.fiit.stuba.sk;
   location / {
        root /home/msevcik/teamweb;
   }
   location ~* /app {
        root /home/ofcsa/prod/public;
        passenger_enabled on;
   }
   location /static {
        autoindex on;
        alias /home/ofcsa/prod/my_app/static/;
    }
}
myapp.conf (END)
```

Obrázok 1: myapp.conf

Tak isto je potrebné získať WSGI Applikáciu pre Django, ktorá sa získava zo súboru passenger_wsgi.py

```
import test_app.wsgi
application = test_app.wsgi.application
passenger_wsgi.py (END)
```

Obrázok 2: passenger_wsgi.py

Na serveri tiež beží databáza PostgreSQL na porte :5432. Takisto tam beží ElasticSearch na porte :9200.

Pre continuous integration server hostí takisto service pre tfs agenta.

Monitorovanie servera

Na monitorovanie servera sa používa systém netdata. Systém monitoruje rôzne prvky servera od CPU cez RAM až po sledovanie výpadkov v komunikácii.

Taktiež je nastavená komunikácia so Slack-om. V prípade, že na serveri dôjde k nejakému výpadku alebo vysokému vyťaženiu niektorého z komponentov, odošle upozornenie na Slack do kanálu alarms.

Na serveri sa takisto sledujú aj chyby/exceptions pomocou systému Rollbar. Tento systém tiež odosiela v prípade zachytenia nejakej chyby pripomienku na Slack. Systém takisto zaznamená celý stack trace aj iné údaje, ako napríklad kde vznikla chyba, z akej IP prišla požiadavka, aký prehliadač bol použitý a koľkokrát táto chyba už nastala.

Správa cloudových služieb

Na správu cloudových služieb sme implementovali funkcionalitu, ktorá používateľovi umožňuje registrovať sa ako poskytovateľ cloudových služieb, registrovať svoje cloudové služby a prezerať si prehľad svojich služieb.

Registrácia poskytovateľa cloudových služieb

Analýza

Používateľ, ktorý prevádzkuje cloudové služby, musí mať možnosť zaregistrovať sa ako prevádzkovateľ cloudovej služby.

Návrh

Pre poskytovateľa cloudovej služby potrebujeme vedieť jeho názov. Jeden používateľ môže byť registrovaný ako poskytovateľ cloudových služieb len raz.

Implementácia

Implementovali sme formulár, v ktorom prihlásený používateľ zadá meno poskytovateľa a potvrdí ho.

Testovanie

Testovanie registrácie poskytovateľa prebieha najmä prostredníctvom jednotkových testov. Jednotkovými testami sa zameriavame na testovanie vyžiadania prihlasovacích údajov pri prístupe na formulár. Testy zahŕňajú správny prípad použitia po prihlásení používateľa, ale aj nesprávne scenáre pri zadaní nesprávnych údajov, nevyplnení polí alebo neprihlásení používateľa.

Registrácia cloudovej služby

Analýza

Používateľ, ktorý prevádzkuje cloudové služby, musí mať možnosť zaregistrovať svoje cloudové služby.

Návrh

Pre cloudovú službu potrebujeme vedieť jej názov, opis, typ služby a typ jej nasadenia. Poskytovateľ cloudovej služby sa získa z prihláseného používateľa.

Implementácia

Implementovali sme formulár, v ktorom prihlásený používateľ zadá všetky potrebné údaje a potvrdí ich.

Testovanie

Testovanie vytvorenia služby prebieha najmä prostredníctvom jednotkových testov. Jednotkovými testami sa zameriavame na testovanie vyžiadania prihlasovacích údajov pri prístupe na formulár. Testy zahŕňajú správny prípad použitia po prihlásení používateľa, ale aj

nesprávne scenáre pri zadaní nesprávnych údajov, nevyplnení polí alebo neprihlásení používateľa.

Prehľad služieb poskytovateľa

Analýza

Používateľ, ktorý prevádzkuje cloudové služby, musí mať možnosť prehliadať si svoje cloudové služby.

Návrh

Pre cloudovú službu potrebujeme v prehľade vidieť jej názov, opis, typ služby a typ jej nasadenia. Poskytovateľ cloudovej služby sa získa z prihláseného používateľa.

Implementácia

Implementovali sme pohľad, v ktorom prihlásený používateľ vidí prehľad svojich registrovaných služieb so všetkými dostupnými údajmi.

Testovanie

Testovanie vytvorenia služby prebieha najmä prostredníctvom jednotkových testov. Jednotkovými testami sa zameriavame na testovanie vyžiadania prihlasovacích údajov pri prístupe na formulár. Testy zahŕňajú správny prípad použitia po prihlásení používateľa, ale aj nesprávne scenáre pri neprihlásení používateľa.

Automatické vytváranie opisu certifikačnej schémy

Opis certifikačných schém je po importovaní zo súboru ďalej nutné upravovať. Po importovaní sa v systéme nachádzajú len schéma, control-y a otázky control-ov. Je teda nutné vygenerovať control objective-y a bezpečnostné atribúty a metriky. Až potom je možné považovať automatické vytváranie schémy za ukončené.

Rozdelenie opisu control-u na opisy control objective-ov

Analýza

Používateľ chce vytvoriť popisy control objective-ov pre importované control-y a nechce ich vytvárať ručne.

Návrh

Prihlásený používateľ musí mať možnosť vybrať si control certifikačnej schémy, ktorý nemá pridelené žiadne control objective-y a dať si vygenerovať ich opisy z opisu control-u.

Implementácia

Proces je implementovaný pomocou dvoch formulárov. V prvom si používateľ vyberie control, pre ktorý chce vygenerovať control objective-y. V druhom sú mu poskytnuté vygenerované opisy pre control objective-y, kde ich môže používateľ ešte upraviť a uložiť. Základným kameňom generovania opisov pre control objective-y je knižnica Stanford

CoreNLP, ktorá slúži na spracovanie prirodzeného textu. Problémom, ktorý sa v tomto kroku rieši, je rozdelenie súvetia na viaceré jednoduché vety.

Testovanie

Testovanie daného procesu sa vykonáva jednotkovými testami, ktoré otestujú formulár definovaný form-om, a view funkciu, ktorá obsahuje logiku za formulárom. Existujú aj malé množstvo integračných testov, ktoré overujú možnosť navigácie v aplikácii pomocou breadcrumbs. Jednotkovými testami je pokrytá aj samostatná funkcionalita rozdeľovania súvetí na jednoduché vety.

Rozdelenie opisu otázok control-ov na opisy control objective-ov

Analýza

Používateľ chce vytvoriť opisy control objective-ov pre importované otázky control-ov automaticky.

Návrh

Prihlásený používateľ musí mať možnosť vybrať si otázku control-u certifikačnej schémy, ktorá nemá pridelené žiadne control objective-y a dať si vygenerovať ich opisy z opisu otázky control-u.

Implementácia

Proces je implementovaný pomocou dvoch formulárov. V prvom si používateľ vyberie otázku control-u, pre ktorú chce vygenerovať control objective-y. V druhom sú mu poskytnuté vygenerované opisy pre control objective-y, kde ich môže ešte upraviť a uložiť. Využíva sa knižnica Stanford CoreNLP, ktorá slúži na spracovanie prirodzeného textu. V prípade otázky dochádza k preusporiadaniu slov, aby vzniknuté jednoduché vety boli oznamovacie.

Testovanie

Testovanie sa vykonáva jednotkovými testami, ktoré otestujú formulár definovaný form-om a view funkciu, ktorá obsahuje logiku za formulárom. Existujú aj malé množstvo integračných testov, ktoré overujú možnosť navigácie v aplikácii pomocou breadcrumbs. Jednotkovými testami je pokrytá aj funkcionalita parsovania otázky a generovanie oznamovacích viet.

Rozdelenie opisu control objective-u na bezpečnostné atribúty a metriky

Analýza

Po vytvorení control objective-ov je potrebné k nim priradiť bezpečnostné atribúty, a metriku ku každému bezpečnostnému atribútu. Používateľ chce pokračovať v importovaní schémy a nechce vytvárať všetky bezpečnostné atribúty a metriky ručne. Preto mu poskytneme možnosť, aby boli nájdené alebo automaticky vygenerované.

Návrh

Prihlásený používateľ musí mať možnosť vybrať si control objective niektorého z control-ov. Tento control objective sa presmeruje na generovanie ďalších jeho atribútov a metrík, ak sám vznikol pomocou generovania.

Implementácia

Proces je implementovaný pomocou už existujúcich obrazoviek, kde používateľ v prehľade control-u vyberie generovaný control objective. Ďalej sa mu zobrazí obrazovka podobná vytváraniu control objective-u. Ak v databáze existujú bezpečnostné atribúty a/alebo metriky, ktoré sú označené ako vyhovujúce pre opis daného control objective-u, sú vyhovujúcimi dátami vyplnené polia s názvom a opisom bezpečnostného atribútu a metriky. Používateľ následne môže navrhnutý bezpečnostný atribút s metrikou uložiť. Ak sa žiadne vyhovujúce bezpečnostné atribúty ani metriky nenašli, na základe opisu vybraného control objective-u sú automaticky vyplnené polia s názvom a opisom bezpečnostného atribútu a metriky.

Testovanie

Testovanie daného procesu sa vykonáva jednotkovými testami, ktoré testujú očakávané správanie NLP knižnice.

NLP

Jednotlivé opisy control-ov a otázok control-ov sú uvedené v komplexných vetách, ktoré je nutné prepísať na jednoduché opisy control objective-ov. Toto je automatizované pomocou knižnice Stanford CoreNLP, ktorá z komplexných viet vytvorí stromovú štruktúru, ktorá sa spracováva. Jednotlivé slová sú ešte upravované knižnicou Pattern do správneho tvaru. Výsledkom je niekoľko nových viet. Ďalším využitím knižníc je parsovanie samotného opisu control objective-u a odporúčanie hodnoty control objective-u, nových bezpečnostných atribútov a metrík.

Analýza

Pri vytváraní opisov control objective-ov je nutné parsovať komplexné vety a prepísať ich na jednoduché automaticky. Pri odporúčaní hodnoty control objective-u, nových bezpečnostných atribútov a metrík je potrebné rozdeliť vetu na správne časti.

Návrh

Treba nasadiť Stanford CoreNLP na server. Je nutné implementovať parsovanie komplexných viet na jednoduché vety a ich úpravu do správneho tvaru. Je potrebné odporúčať správny tvar hodnoty control objective-u, bezpečnostného atribútu a metriky na základe opisu control objective-u.

Implementácia

Modul je implementovaný vo viacerých funkciách, ktoré využívajú stromovú štruktúru parsovanej vety. Najskôr sa zo stromovej štruktúry vygenerovanej knižnicou Stanford CoreNLP vytvorí naša stromová štruktúra, do ktorej si už ukladáme potrebné premenné pri

prechode stromom. Pri generovaní komplexných viet na jednoduché vety sa tento strom predspracováva a následne je implementované prechádzanie do hĺbky. Týmto prechodom sa vygenerujú jednoduché vety, ktoré sú ešte ďalej upravené. Jednotlivé tvary slov sú upravené do jednotného tvaru pomocou knižnice Pattern. Pri parsovaní jednoduchej vety opisu control objective-u sa prechádza strom na základe tag-ov vrcholov stromu, aby sa vždy získala správna časť vety.

Testovanie

Testovanie celého modulu sa vykonáva jednotkovými testami, ktoré testujú jednotlivé funkcie. Pre každú funkciu sú testami pokryté ľubovoľné vstupy a typy viet, keďže používateľ môže zadávať ľubovoľný text do opisu control-u, otázky control-u aj control objective-u.

Inštalačná príručka

Systém pozostáva z aplikácie napísanej v Python framework-u Django. Aplikácia využíva niekoľko Python knižníc, ktoré sú obsiahnuté v súbore *requirements.txt*. Dáta sa ukladajú do relačnej databázy, počas vývoja sa použila databáza PostgreSQL, ale aplikácia by mala fungovať aj nad inými SQL databázami. Ako server sa použil NGINX server s Passenger-om, ktorý zabezpečuje vyvažovanie záťaže.

Na zjednodušovanie opisu control-ov a generovanie control objective-ov využívame knižnicu Stanford CoreNLP napísanú v Jave. Rovnako sa pri tom využíva ElasticSearch, ktorý sa však používa aj na iné funkcie aplikácie.

Počas vývoja sa na zachytávanie chýb využíval systém Rollbar, ktorý je odporúčaný používať naďalej, vzhľadom na to, že poskytuje najlepšie hlásenie chýb pre vývojárov.

Potrebné požiadavky pred inštaláciou

Nutne požadované podmienky, ktoré musíte mať pred inštaláciou našej aplikácie sú:

- Python 3
- requirements.txt
- server
- Stanford CoreNLP
- SQL Databaza
- ElasticSearch

Ďalej odporúčané, ale nevyžadované:

- passenger
- rollbar

Inštalačné kroky

V tejto časti predpokladáme, že sa aplikácie inštaluje na operačný systém Linux. Nasledujúce kroky sa môžu, ale nemusia líšiť pri inštalovaní na platformu Windows.

1. Nainštalovať Python 3 podľa inštrukcií na: https://www.python.org/
2. Nainštalovať a spustiť SQL databázu, odporúčame PostgreSQL: https://www.postgresql.org/
3. Nainštalovať requirements.txt pomocou: *pip install -r /path/to/requirements.txt*
4. Nainštalovať webový server, odporúčame Nginx: https://www.nginx.com/
    a. [Nepovinné] Nainštalovať Passenger: https://www.phusionpassenger.com/library/config/nginx/intro.html
5. Stiahnuť a spustiť Standford CoreNLP: https://stanfordnlp.github.io/CoreNLP/
6. Stiahnuť a spustiť ElasticSearch: https://www.elastic.co/

7. [Nepovinné] Nainštalovať Rollbar: https://rollbar.com/
8. Rozbaliť/Stiahnuť súbory aplikácie.
9. Nakonfigurovať jednotlivé súčasti podľa časti Konfigurácia alebo podľa vlastného uváženia.

Konfigurácia

Konfigurovať je hlavne potrebné Nginx server. V adresári */etc/nginx/sites-enabled* obsah súboru vyzerá nasledovne:

```
server {

  passenger_python /usr/bin/python3.6;

  listen 80;

  server_name team12-17.studenti.fiit.stuba.sk;


  location = /netdata {

      return 301 /netdata/;

  }

  location ~ /netdata/(?<ndpath>.*) {

      proxy_redirect off;

      proxy_set_header Host $host;


      proxy_set_header X-Forwarded-Host $host;

      proxy_set_header X-Forwarded-Server $host;

      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

      proxy_http_version 1.1;

      proxy_pass_request_headers on;

      proxy_set_header Connection "keep-alive";

      proxy_store off;

      proxy_pass http://netdata/$ndpath$is_args$args;
```

```
        gzip on;
        gzip_proxied any;
        gzip_types *;
        auth_basic "Protected";
        auth_basic_user_file passwords;
    }
    location ~* /appdev {
        root /home/ofcsa/myagent/_work/6/s/public;
        passenger_enabled on;
    }
    location / {
        root /home/ofcsa/myagent/_work/3/s;
    }
    location ~* /app {
        root /home/ofcsa/prod/public;
        passenger_enabled on;
    }
    location /static {
        autoindex on;
        alias /home/ofcsa/prod/my_app/static/;
     }


    listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/team12-17.studenti.fiit.stuba.sk/fullchain.pem; # managed by Certbot
```

```
ssl_certificate_key /etc/letsencrypt/live/team12-17.studenti.fiit.stuba.sk/privkey.pem; # managed by Certbot

    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


    if ($scheme != "https") {

        return 301 https://$host$request_uri;

    } # managed by Certbot


}
```

Ďalšou konfiguráciou je konfigurácia aplikácie samotnej, ktorú treba nakonfigurovať pomocou vytvorenia súboru settings.ini skopírovaním a odstránením prípony .template zo súboru settings.ini.template v adresári test_app/settings/ našej aplikácie.

V tomto súbore treba nastaviť názov databázy, používateľa a jeho heslo. Ostatné nastavenia sú voliteľné a sú na uvážení používateľa.

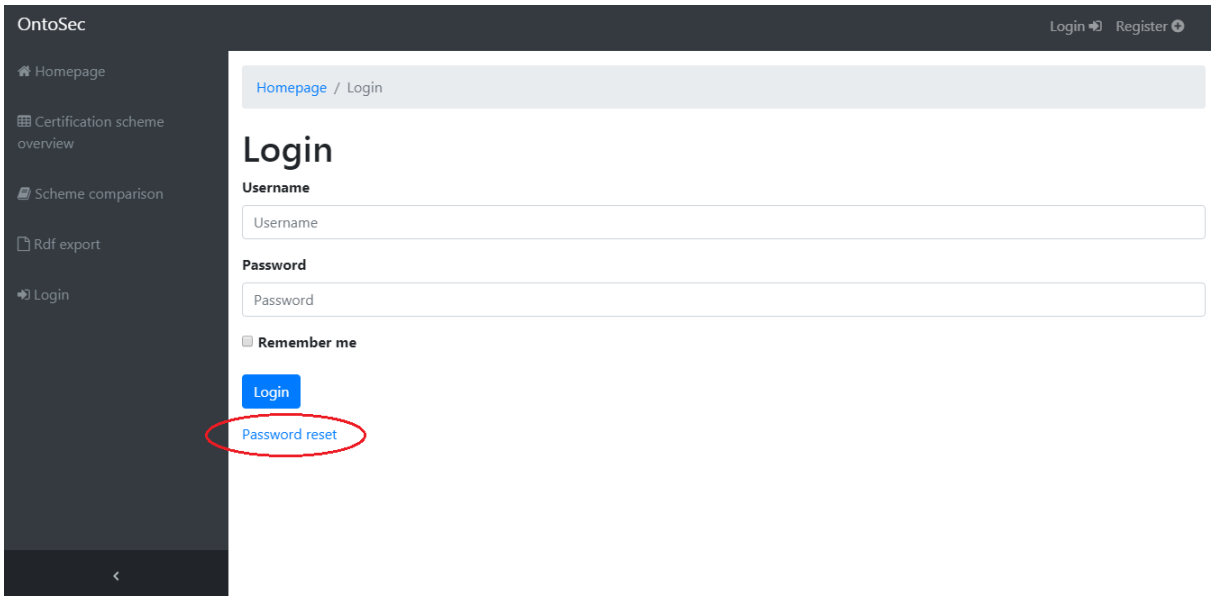
Standford NLP je najlepšie spúšťať:

java -mx4g -cp "" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -annotators "tokenize,ssplit,pos,lemma,parse,sentiment" -port 9685 -timeout 30000*

Používateľská príručka

Obnovenie hesla

Používateľom s kontom na stránke sa môže stať, že zabudnú svoje heslo alebo meno. Ak sa to stane, môžu vybrať možnosť pre obnovu hesla na stránke pre prihlásenie (Obr. 1).

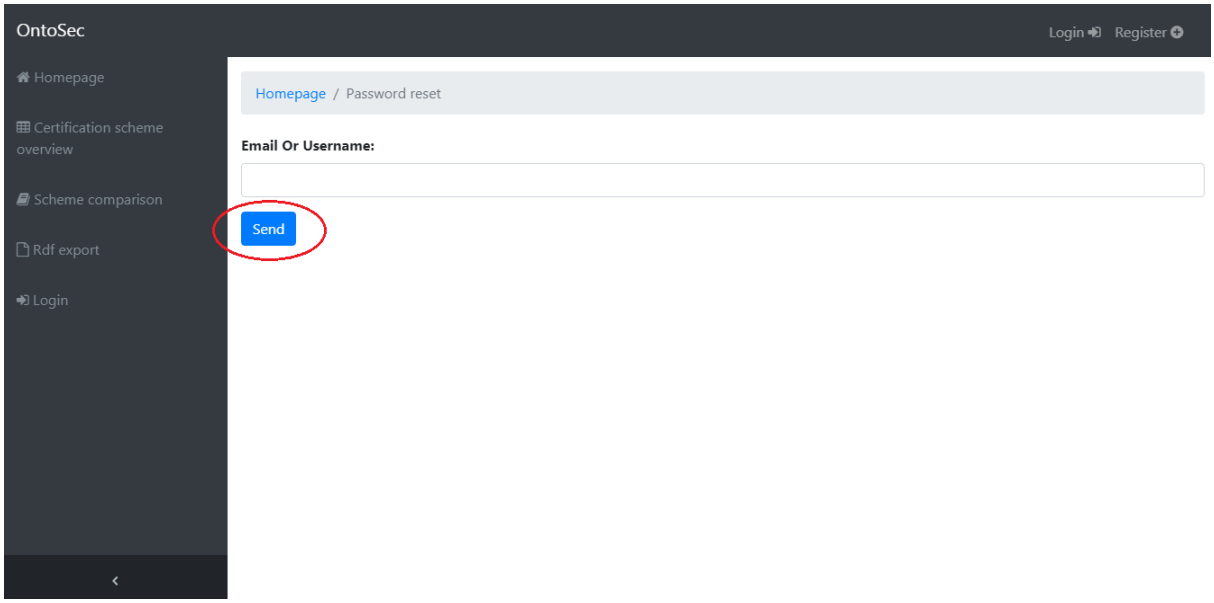


Obrázok 1. Prihlásenie.

Následne používateľ zadá svoje meno alebo heslo a klikne na tlačidlo **Send** (Odoslať) (Obr. 2). Môže nastať 5 situácií:

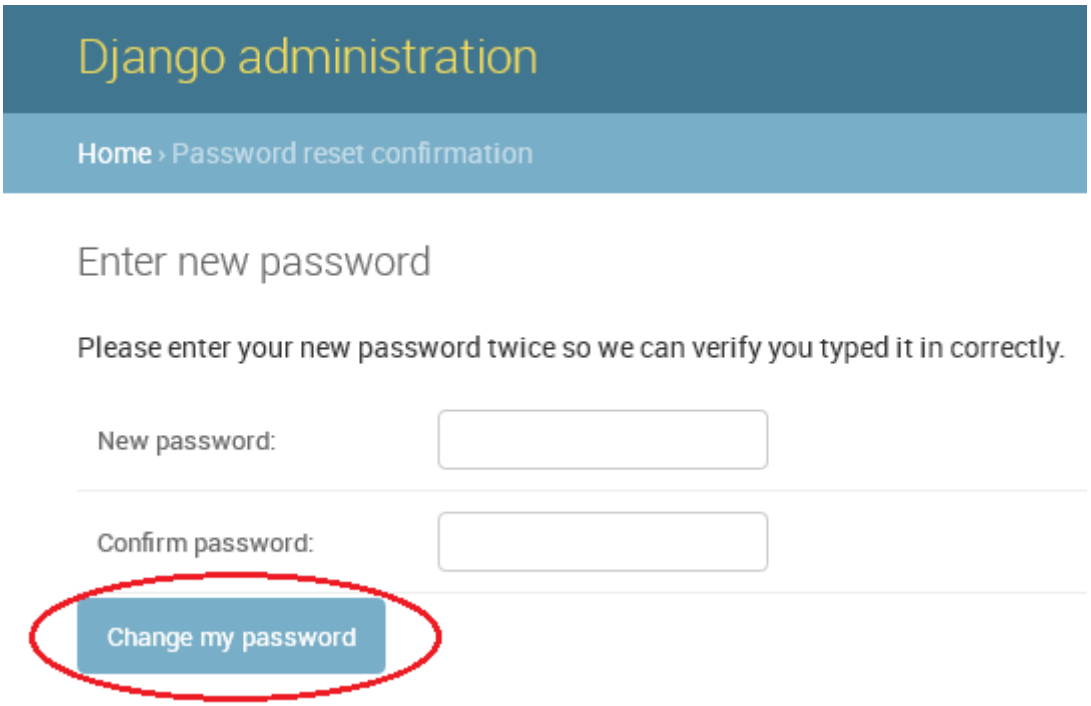
1. používateľ zadá správne prihlasovacie meno,
2. používateľ zadá správnu e-mail adresu,
3. používateľ zadá nesprávne prihlasovacie meno,
4. používateľ zadá nesprávnu e-mail adresu,
5. používateľ nezadá nič.

Pri každej z týchto situácií bude používateľovi poskytnutá informácia. Ak zadal správne informácie, bude vyzvaný, aby si pozrel svoju e-mailovú schránku, kam mu má prísť správa s odkazom na vytvorenie nového hesla.



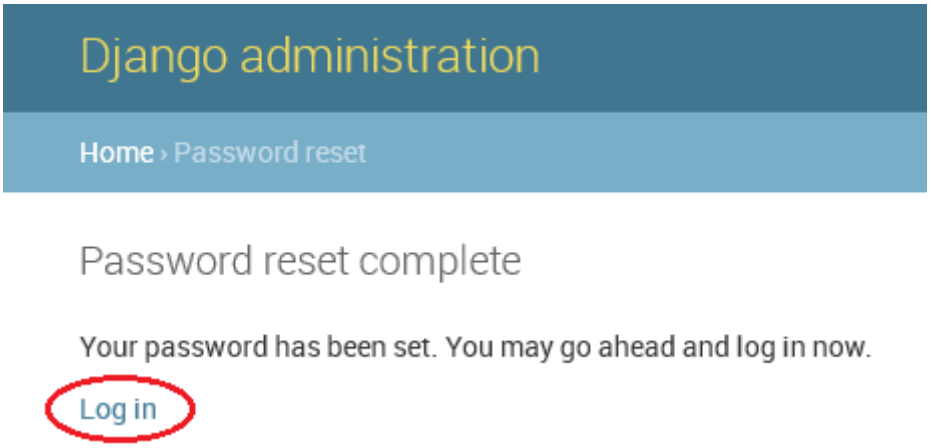
Obrázok 2. Obnova hesla.

Po otvorení odkazu z e-mailu, bude používateľovi umožnené vytvorenie nového hesla (Obr. 3).



Obrázok 3. Vytvorenie nového hesla.

Po korektnom vytvorení nového hesla sa používateľ bude môcť prihlásiť pomocou nového hesla (Obr. 4).



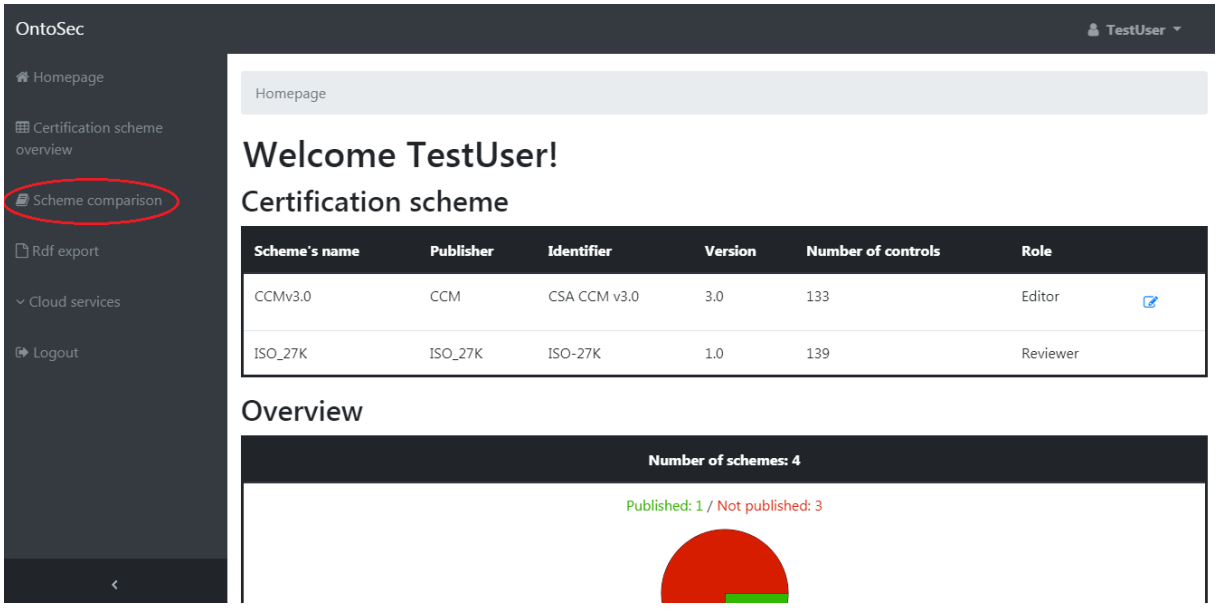
Obrázok 4. Dokončená obnova hesla.

Porovnanie certifikačných schém

Porovnanie certifikačných schém je dostupné v menu aplikácie prihláseným aj neprihláseným používateľom. Schémy sa porovnávajú na základe bezpečnostných atribútov.

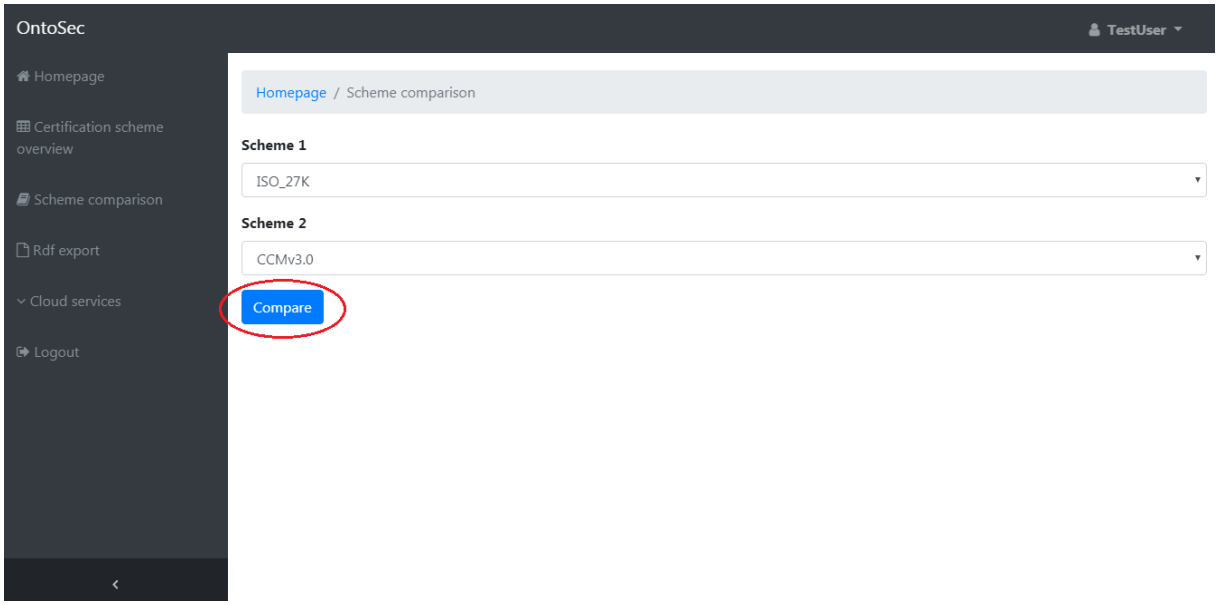
Výber schém na porovnanie

Používateľ v ponuke vyberie možnosť porovnanie schém v menu (Obr. 1).



Obrázok 1. Výber porovnania schém v menu.

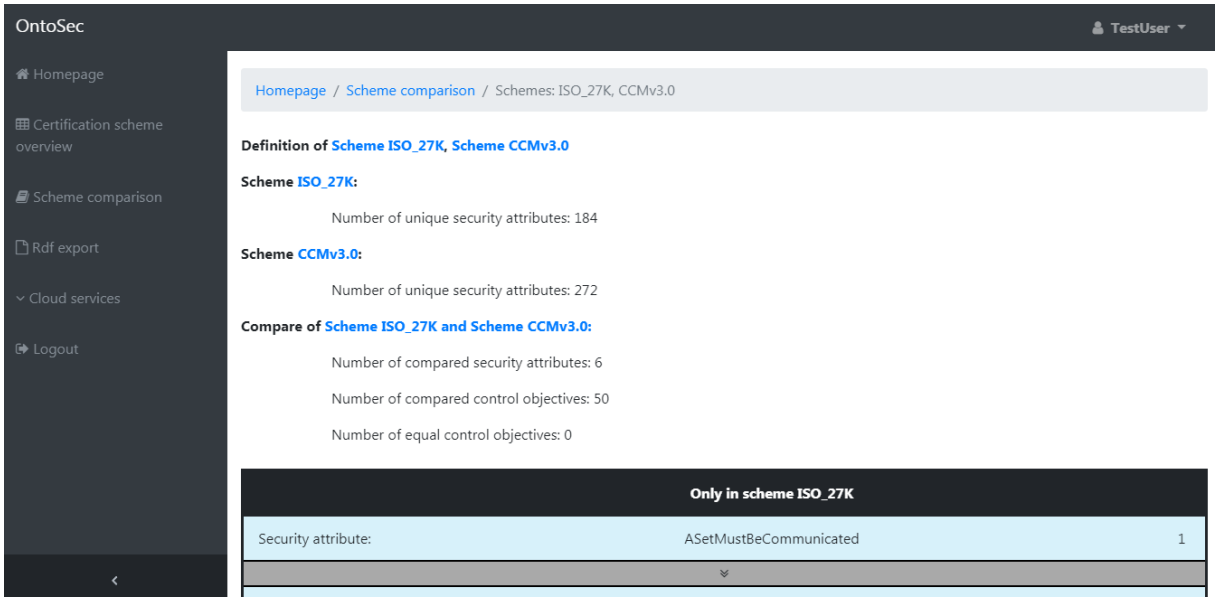
Následne používateľ vyberie schémy, ktoré chce medzi sebou porovnať a klikne na tlačidlo **Compare** (Porovnať) (Obr. 2).



Obrázok 2. Výber schém na porovnanie.

Porovnanie schém

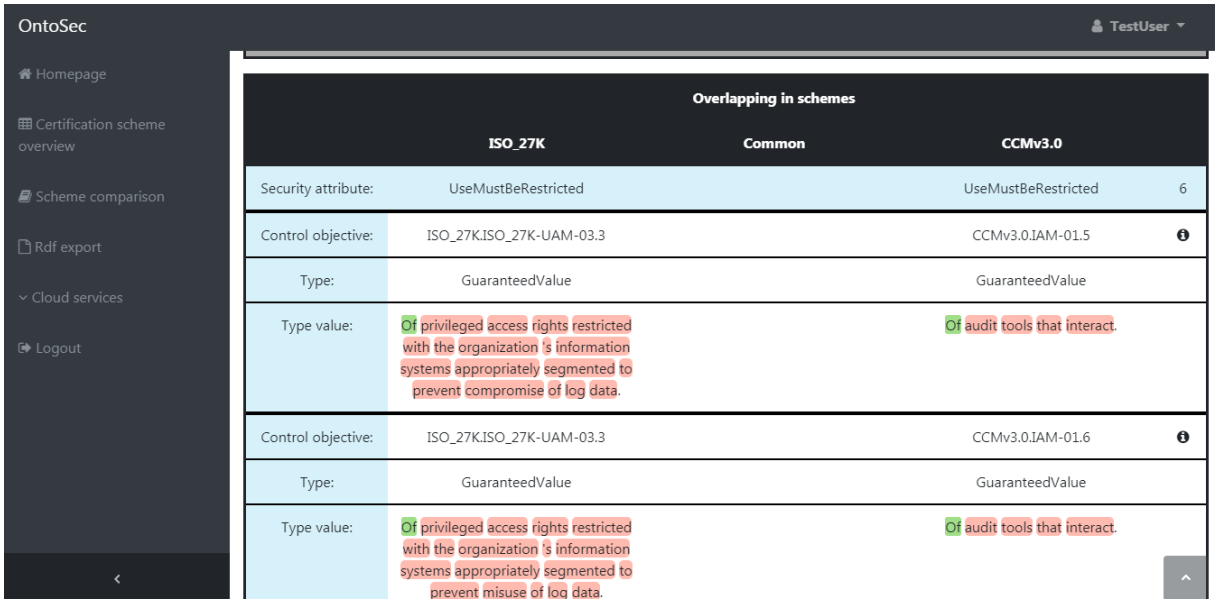
Porovnanie schém je rozdelené do troch tabuliek. Nad nimi sa nachádza zhrnutie porovnania. V prvej a druhej tabuľke sa nachádzajú bezpečnostné atribúty, ktoré sa nachádzajú len v jednej, alebo v druhej schéme (Obr. 3).



Obrázok 3. Zhrnutie a prvá tabuľka porovnania.

V poslednej tabuľke sú zobrazené control objective-y zoskupené podľa bezpečnostných atribútov, ktoré sa istým spôsobom prekrývajú v oboch schémach. Každý riadok prekrývajúcich sa bezpečnostných atribútov sa dá rozkliknúť pre zobrazenie podrobností prekrytia. Používateľ tak dokáže zistiť, v ktorých control objective-och sa prekrývajú.

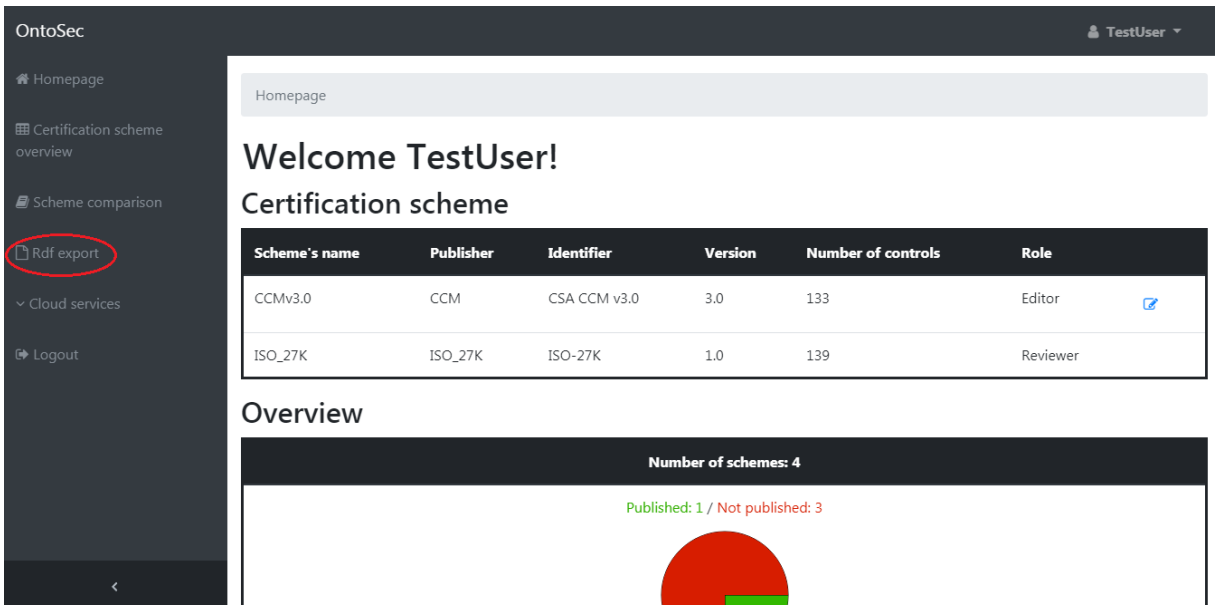
V tabuľke sa porovnajú hodnoty control objective-ov po slovách. Ak sú dve vety rovnaké, nachádzajú sa v strednom stĺpci s názvom Common.



Obrázok 4. Prekrývajúce sa bezpečnostné atribúty.

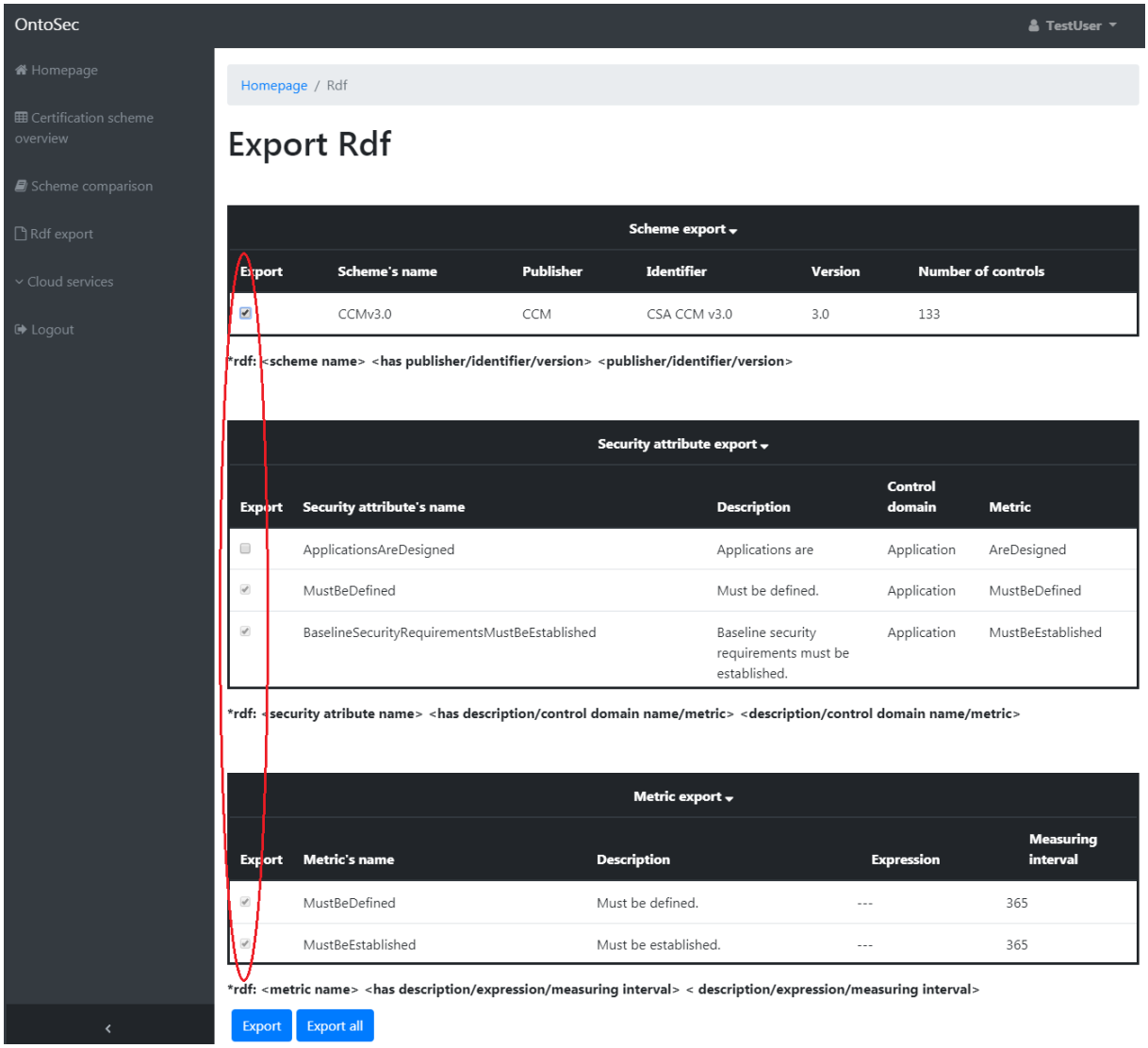
RDF export

Ak chce používateľ vygenerovať súbor, v ktorom budú certifikačné schémy, jej control-y, control objective-y, bezpečnostné atribúty, metriky a ich ostatné parametre, musí sa ako prvý krok navigovať do menu a v ňom do **Rdf export** (Obr. 1). Na stránku Rdf exportu môže pristupovať len prihlásený používateľ.



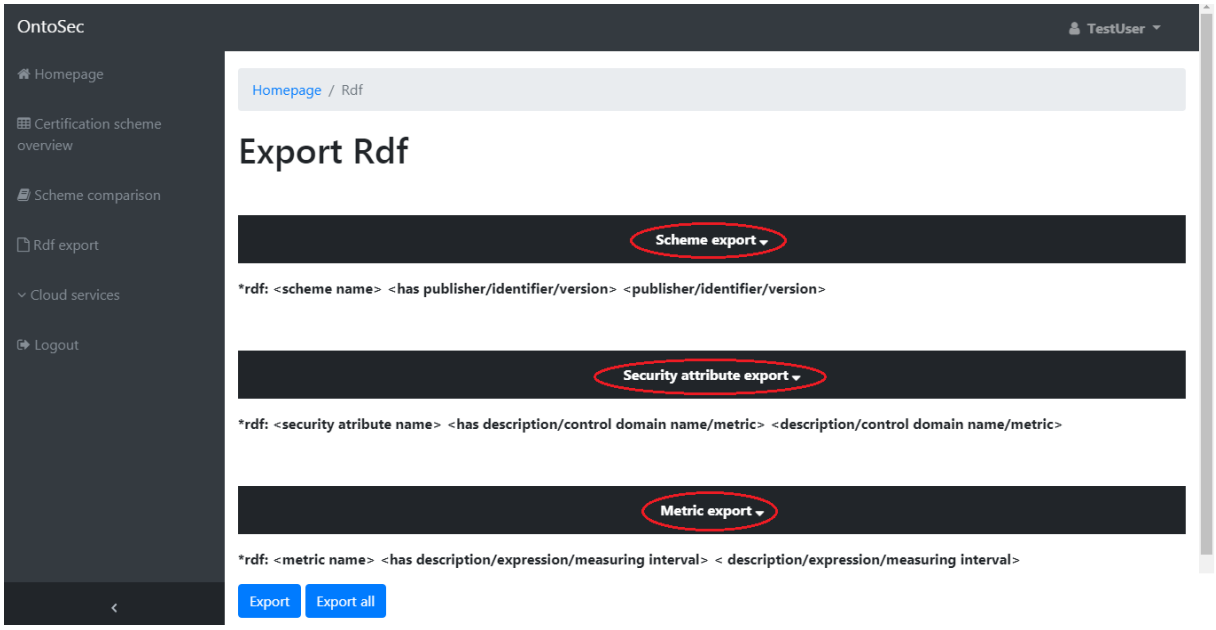
Obrázok 1. Rdf export v menu.

Po zobrazení stránky pre Rdf export sa používateľovi zobrazia 3 tabuľky. V prvej je zoznam schém, ktorých je vlastník, môže danú schému recenzovať alebo editovať, v druhej sú všetky bezpečnostné atribúty a v tretej všetky metriky (Obr. 2). Pre schémy z prvej tabuľky platí, že sa vždy exportujú aj príslušné control-y, control objective-y, bezpečnostné atribúty aj metriky vzhľadom k vybranej schéme. Pre bezpečnostné atribúty z druhej tabuľky sa vždy exportujú aj údaje a metriky súvisiace s daným bezpečnostným atribútom. Pre metriky z tretej tabuľky sa napokon exportujú iba údaje danej metriky.



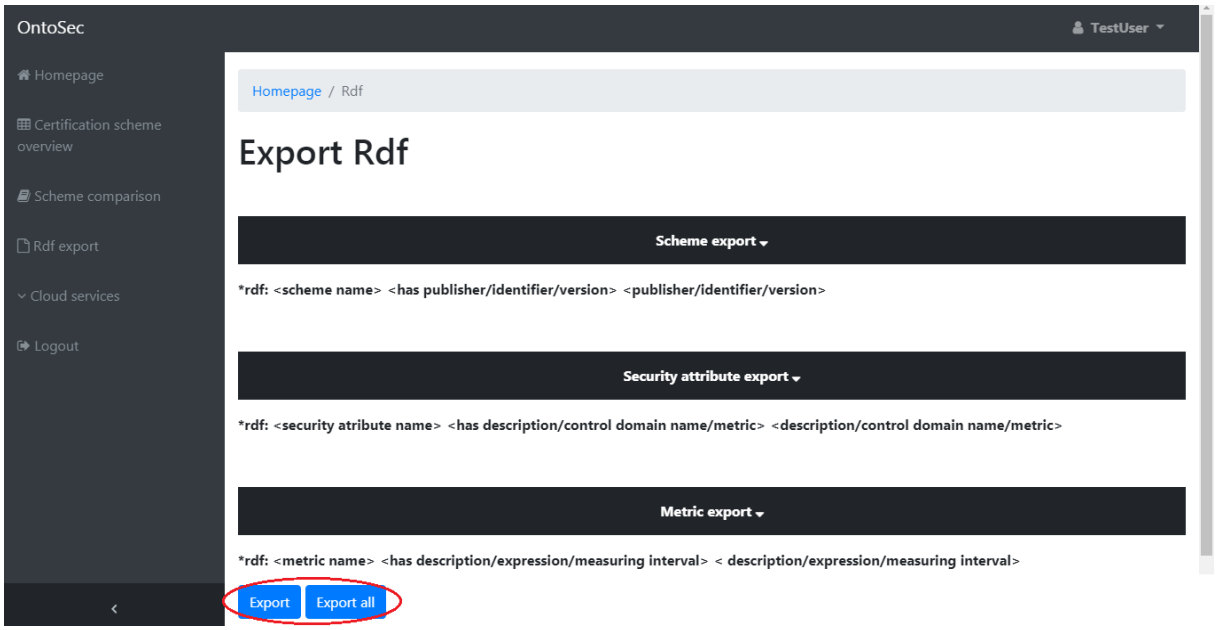
Obrázok 2. Voľba na generovanie Rdf.

Pre lepší prehľad je možné jednotlivé tabuľky zrolovať a vyrolovať kliknutím na názov tabuľky (Obr. 3).



Obrázok 3. Vyrolovanie a zrolovanie tabuliek.

Po kliknutí na tlačidlo **Export** (Obr. 4) sa vygeneruje súbor vo formáte rdf, v ktorom sa budú nachádzať dáta v podobe, ako je v príklade (Príklad 1). V prípade, že používateľ chce exportovať všetky záznamy, klikne na tlačidlo **Export all** (Exportovať všetky).



Obrázok 4. Tlačidlá Export a Export all.

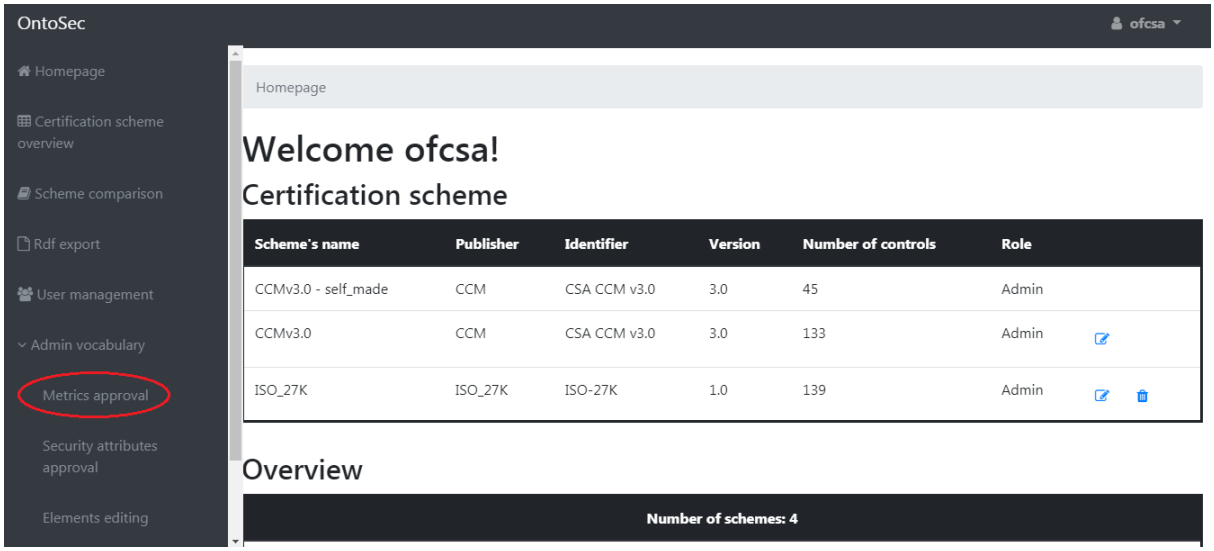
<http://fiit.stuba.sk/eusec#securityAttribute/co2> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#securityAttribute/co2> .
<http://fiit.stuba.sk/eusec#metric/metric> <http://fiit.stuba.sk/eusec#hasMeasuringInterval> <http://fiit.stuba.sk/eusec#metric/365> .
<http://fiit.stuba.sk/eusec#certificationScheme/Profi> <http://fiit.stuba.sk/eusec#hasVersion>
<http://fiit.stuba.sk/eusec#certificationScheme/1.0> .
<http://fiit.stuba.sk/eusec#metric/m1> <http://fiit.stuba.sk/eusec#hasExpression> <http://fiit.stuba.sk/eusec#metric/aaa> .
<http://fiit.stuba.sk/eusec#controlObjective/identt> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#controlObjective/descc> .
<http://fiit.stuba.sk/eusec#securityAttribute/co1> <http://fiit.stuba.sk/eusec#hasMetric> <http://fiit.stuba.sk/eusec#metric/m1> .
<http://fiit.stuba.sk/eusec#certificationScheme/Profi> <http://fiit.stuba.sk/eusec#hasControl> <http://fiit.stuba.sk/eusec#control/data> .
<http://fiit.stuba.sk/eusec#controlDomain/Application> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#controlDomain/Application+%26+Interface+Security> .
<http://fiit.stuba.sk/eusec#controlObjective/identt> <http://fiit.stuba.sk/eusec#hasType>
<http://fiit.stuba.sk/eusec#controlObjectiveType/GuaranteedValue> .
<http://fiit.stuba.sk/eusec#metric/m1> <http://fiit.stuba.sk/eusec#hasDescription> <http://fiit.stuba.sk/eusec#metric/m2> .
<http://fiit.stuba.sk/eusec#securityAttribute/co2> <http://fiit.stuba.sk/eusec#hasMetric> <http://fiit.stuba.sk/eusec#metric/metric> .
<http://fiit.stuba.sk/eusec#controlObjective/identt> <http://fiit.stuba.sk/eusec#refersTo>
<http://fiit.stuba.sk/eusec#securityAttribute/co1> .
<http://fiit.stuba.sk/eusec#controlObjective/co2> <http://fiit.stuba.sk/eusec#hasEvaluationInterval>
<http://fiit.stuba.sk/eusec#controlObjective/1> .
<http://fiit.stuba.sk/eusec#certificationScheme/Amate> <http://fiit.stuba.sk/eusec#hasIdentifier>
<http://fiit.stuba.sk/eusec#certificationScheme/AT> .
<http://fiit.stuba.sk/eusec#certificationScheme/Amate> <http://fiit.stuba.sk/eusec#hasPublisher>
<http://fiit.stuba.sk/eusec#certificationScheme/FIIT> .
<http://fiit.stuba.sk/eusec#certificationScheme/Profi> <http://fiit.stuba.sk/eusec#hasPublisher>
<http://fiit.stuba.sk/eusec#certificationScheme/FIIT> .
<http://fiit.stuba.sk/eusec#certificationScheme/Amate> <http://fiit.stuba.sk/eusec#hasVersion>
<http://fiit.stuba.sk/eusec#certificationScheme/1.0> .
<http://fiit.stuba.sk/eusec#securityAttribute/co2> <http://fiit.stuba.sk/eusec#hasDomain>
<http://fiit.stuba.sk/eusec#controlDomain/Application> .
<http://fiit.stuba.sk/eusec#metric/metric> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#metric/metric+description> .
<http://fiit.stuba.sk/eusec#controlObjective/co2> <http://fiit.stuba.sk/eusec#hasValue> <http://fiit.stuba.sk/eusec#controlObjective/1> .
<http://fiit.stuba.sk/eusec#control/data> <http://fiit.stuba.sk/eusec#hasObjective> <http://fiit.stuba.sk/eusec#controlObjective/co2> .
<http://fiit.stuba.sk/eusec#securityAttribute/co1> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#securityAttribute/co1> .
<http://fiit.stuba.sk/eusec#securityAttribute/co1> <http://fiit.stuba.sk/eusec#hasDomain>
<http://fiit.stuba.sk/eusec#controlDomain/Application> .
<http://fiit.stuba.sk/eusec#controlObjective/co2> <http://fiit.stuba.sk/eusec#hasDescription>
<http://fiit.stuba.sk/eusec#controlObjective/co2> .
<http://fiit.stuba.sk/eusec#metric/metric> <http://fiit.stuba.sk/eusec#hasExpression> <http://fiit.stuba.sk/eusec#metric/radio+express> .
<http://fiit.stuba.sk/eusec#controlObjective/identt> <http://fiit.stuba.sk/eusec#hasEvaluationInterval>
<http://fiit.stuba.sk/eusec#controlObjective/8> .
<http://fiit.stuba.sk/eusec#control/data> <http://fiit.stuba.sk/eusec#hasText>
<http://fiit.stuba.sk/eusec#control/Do+you+have+fence+around+your+buildings+with+servers%3F> .
<http://fiit.stuba.sk/eusec#control/data> <http://fiit.stuba.sk/eusec#hasObjective> <http://fiit.stuba.sk/eusec#controlObjective/identt> .
<http://fiit.stuba.sk/eusec#controlObjective/identt> <http://fiit.stuba.sk/eusec#hasValue>
<http://fiit.stuba.sk/eusec#controlObjective/5> .
<http://fiit.stuba.sk/eusec#control/data> <http://fiit.stuba.sk/eusec#hasDescription> <http://fiit.stuba.sk/eusec#control/Fence> .
<http://fiit.stuba.sk/eusec#controlObjective/co2> <http://fiit.stuba.sk/eusec#hasType>
<http://fiit.stuba.sk/eusec#controlObjectiveType/GuaranteedValue> .
<http://fiit.stuba.sk/eusec#controlObjective/co2> <http://fiit.stuba.sk/eusec#refersTo> <http://fiit.stuba.sk/eusec#securityAttribute/co2>
.
<http://fiit.stuba.sk/eusec#certificationScheme/Profi> <http://fiit.stuba.sk/eusec#hasIdentifier>
<http://fiit.stuba.sk/eusec#certificationScheme/PF> .
<http://fiit.stuba.sk/eusec#metric/m1> <http://fiit.stuba.sk/eusec#hasMeasuringInterval> <http://fiit.stuba.sk/eusec#metric/123> .

Príklad 1. Príklad exportovaného rdf súboru.

Schvaľovanie metrík a bezpečnostných atribútov

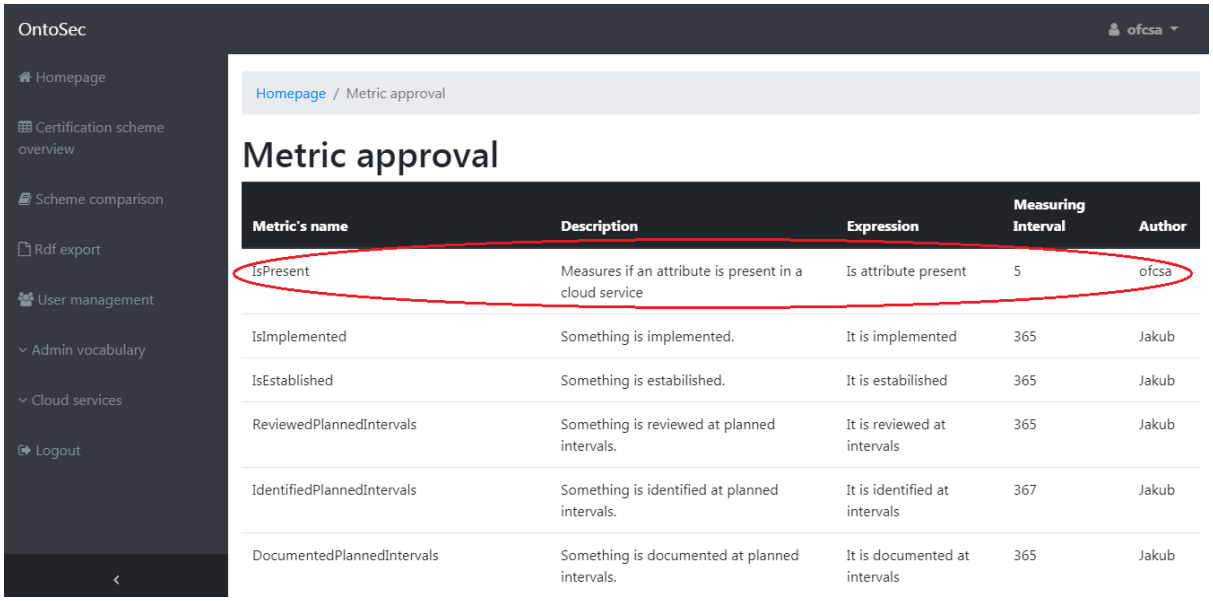
Táto používateľská príručka opisuje návod, ako môže správca schvaľovať, zamietať alebo pozmeniť novovytvorené metriky alebo bezpečnostné atribúty. Postup je veľmi podobný, preto bude opisovaná len možnosť pre metriky.

Správca si vyberie možnosť pre **Metrics approval** (Schvaľovanie metrík) v menu (Obr. 1).



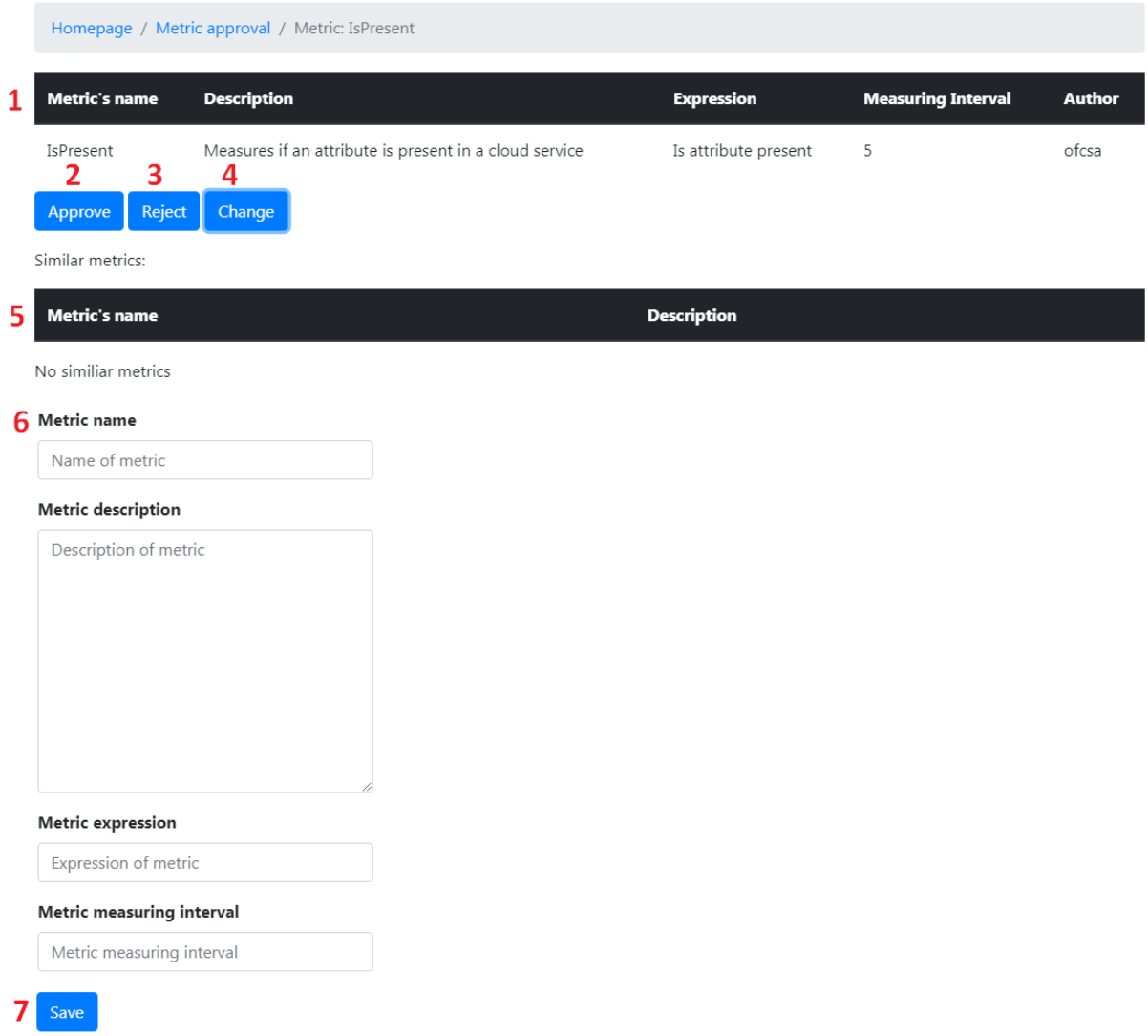
Obrázok 1. Domovská stránka.

Ďalej si vyberie požadovaneú metriku na schvaľovanie (Obr. 2).



Obrázok 2. Výber metriky.

Po vybraní metriky sa mu zobrazí okno, v ktorom môže metriku schváliť, zamietnuť alebo zmeniť (Obr. 3).



Obrázok 3. Vybraná metrika.

1. Opis vybranej metriky.
2. Tlačidlo pre schvaľovanie metriky. Stlačením sa metrika schváli.
3. Tlačidlo pre zamietnutie metriky. Stlačením sa metrika zamietne.
4. Tlačidlo pre zmenu. Stlačením sa zobrazuje/schováva formulár pre zmenu atribútov (6).
5. Zoznam podobných metrík.
6. Formulár pre zmenu atribútov metriky.
7. Tlačidlo pre uloženie zmien. Stlačením sa zmenia atribúty vybranej metriky na zadané.

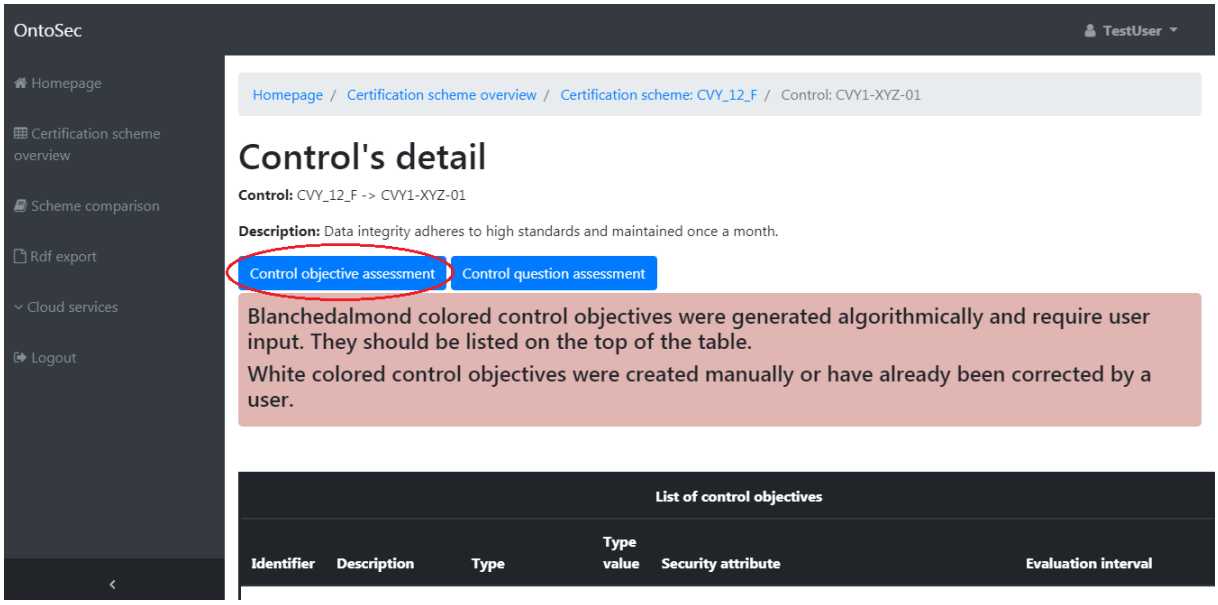
Schvaľovanie schém

Pre používateľov a auditorov je potrebné, aby boli schémy v systéme korektné a bolo tak možné ich porovnanie, ako aj vkladanie nových schém na základe už existujúcich schém. Aby bolo toto možné, je najprv nutné, aby boli jednotlivé elementy korektné, teda schválené

určeným používateľom. Až potom môže byť schválená celá schéma, ktorá tieto control-y a control objective-y obsahuje.

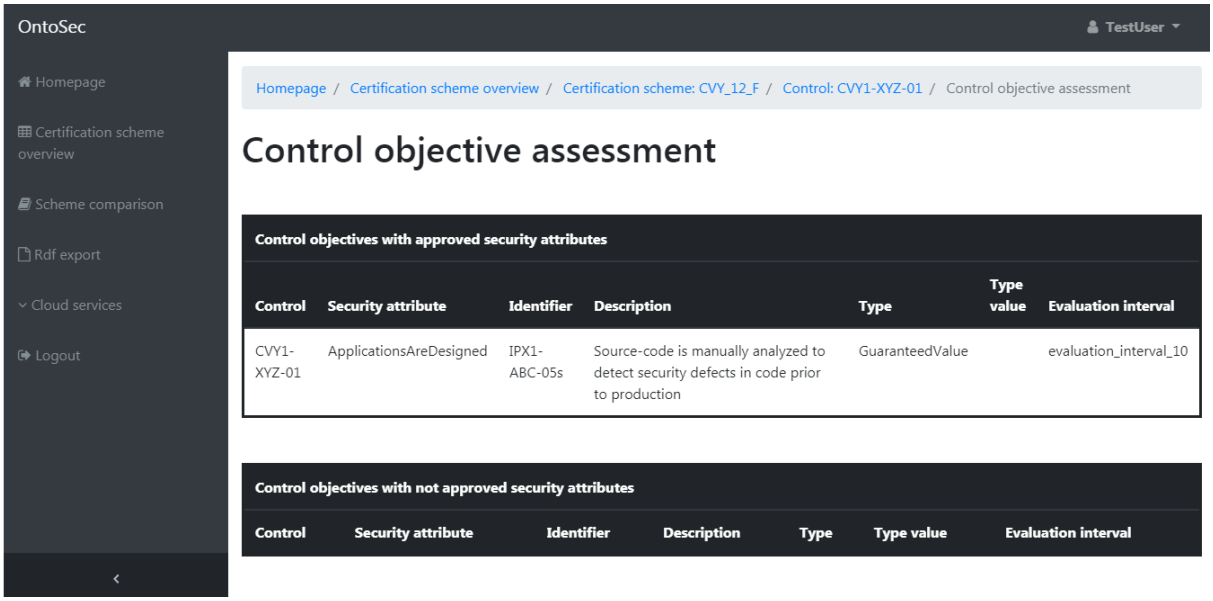
Proces schválenia control objective-u

Schváliť schému je možné iba vtedy, ak obsahuje iba schválené alebo zamietnuté control-y, pričom každý control musí mať schválené všetky control objective-y a otázky control-ov. Na obrázku (Obr. 1) je ukázaná obrazovka, v ktorej sa po stlačení tlačidla **Control objective assessment** (Posúdenie control objective-u) zobrazí obrazovka znázornená na Obrázku 2.



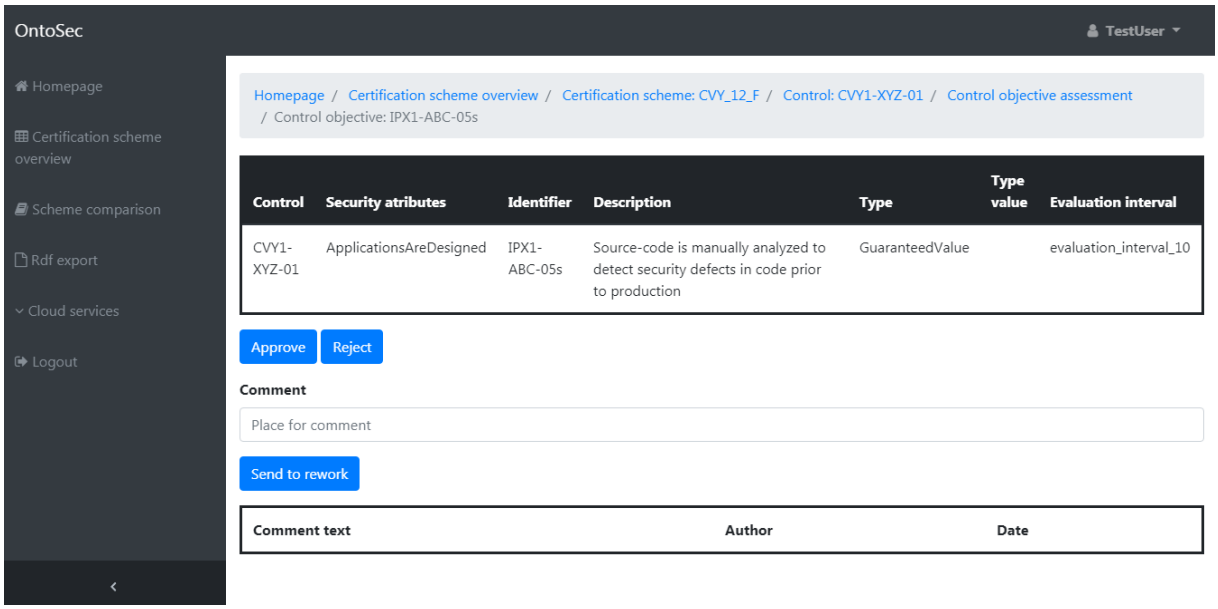
Obrázok 1. Detail control-u.

Na obrazovke z Obrázku 2 je možné kliknúť na jednotlivé control objective-y, ktoré sa nachádzajú v prvej tabuľke s názvom **Control objectives with approved security attributes** (Control objective-y so schválenými bezpečnostnými atribútmi). Po kliknutí na konkrétny control objective sa používateľovi zobrazí obrazovka znázornená na Obrázku 3.



Obrázok 2. Posúdenie control objective-u.

Na obrazovke z Obrázku 3 je možné vidieť control objective zvolený z predchádzajúcej obrazovky. V tejto obrazovke je možné control objective schváliť stlačením tlačidla **Approve** (Schváliť), zamietnuť stlačením tlačidla **Reject** (Zamietnuť) alebo ho poslať pracovníkovi späť na prepracovanie stlačením tlačidla **Send to rework** (Poslať na prepracovanie). V prípade, ak bola zvolená možnosť Send to rework, je nutné zadať komentár, aby bolo pracovníkovi jasné, prečo bol control objective neschválený a mohol ho tak prerobiť na korektný control objective.

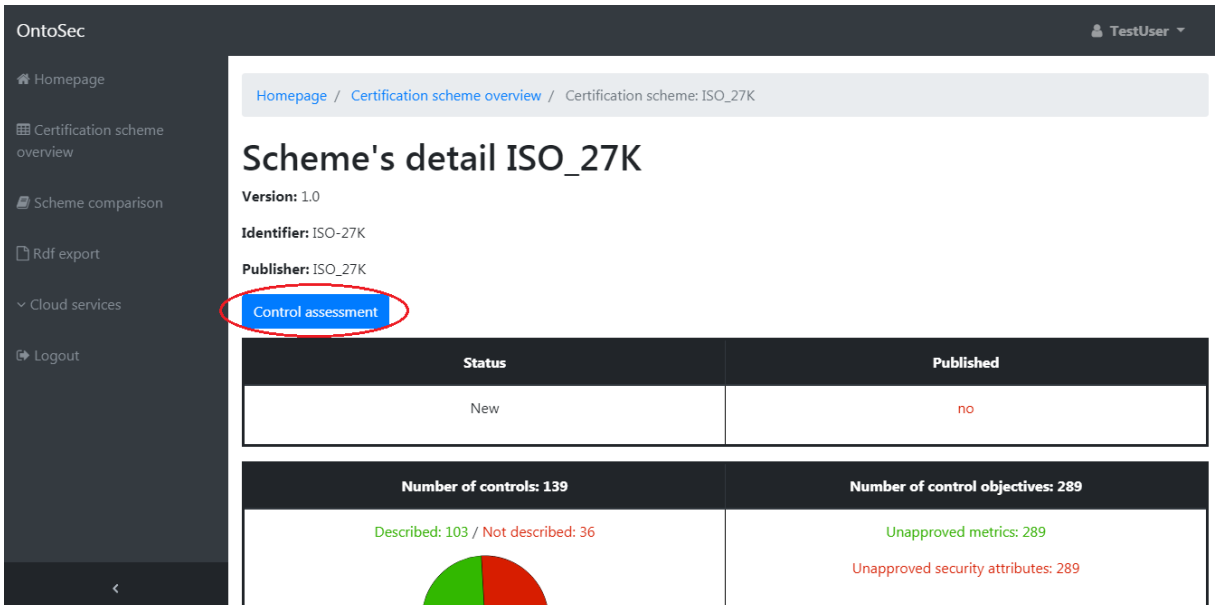


Obrázok 3. Vybraný control objective.

Ak chce používateľ posúdiť otázky control-ov, na obrazovke z Obrázku 1 klikne na tlačidlo **Control question assessment** (Posúdenie otázok control-u). Ďalej je postup rovnaký ako pri posudzovaní control objective-ov.

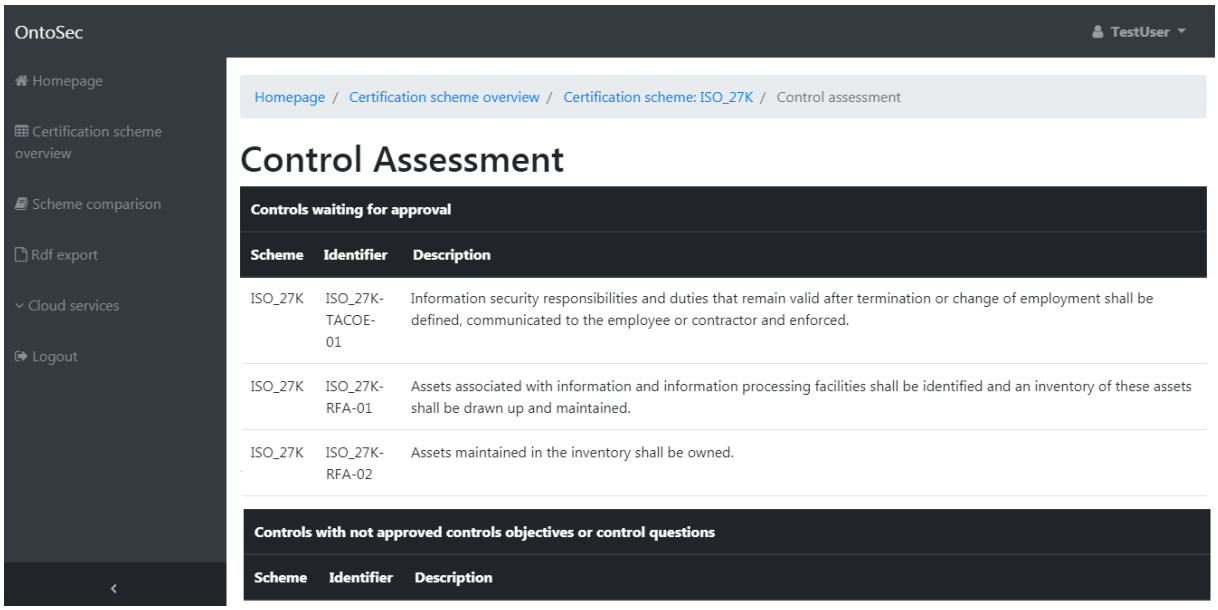
Proces schválenia control-u

V prípade, že všetky control objective-y boli schválené alebo zamietnuté, je možné schváliť jednotlivé control-y schémy. Toto je umožnené v obrazovke Scheme's detail (Detail schémy) po stlačení tlačidla **Control assessment** (Posúdenie control-u) z Obrázku 4.



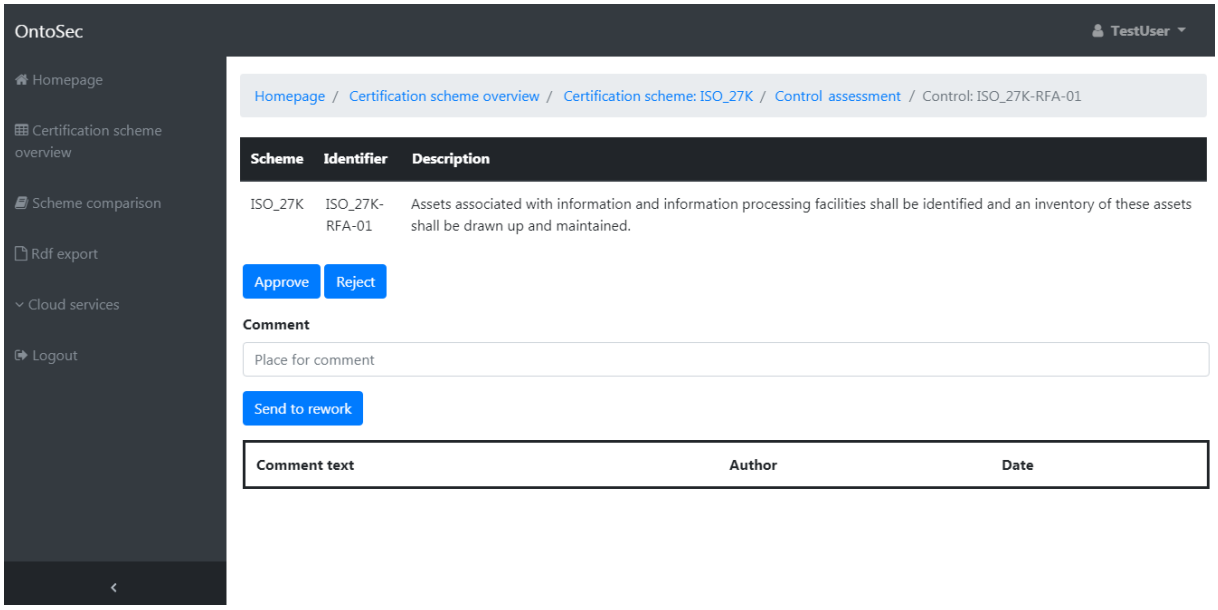
Obrázok 4. Detail schémy.

Používateľovi sa následne zobrazí obrazovka so zoznamom control-ov rozdelených do dvoch tabuliek. Po kliknutí na konkrétny control z prvej tabuľky s názvom **Controls waiting for approval** (Control-y čakajúce na schválenie) sa používateľovi zobrazí obrazovka znázornená na Obrázku 5.



Obrázok 5. Posúdenie control-u.

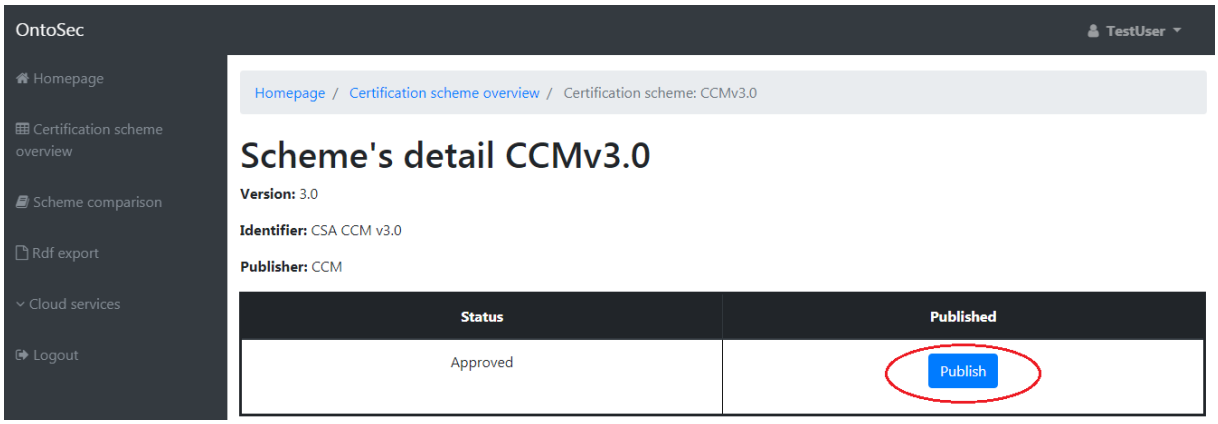
Rovnako ako pri schvaľovaní control objective-u, aj pri schválení control-u sú 3 možnosti, a tými sú schválenie, zamietnutie a poslanie späť pracovníkovi na prerobenie, ako je možné vidieť na Obrázku 6.



Obrázok 6. Vybraný control na posúdenie.

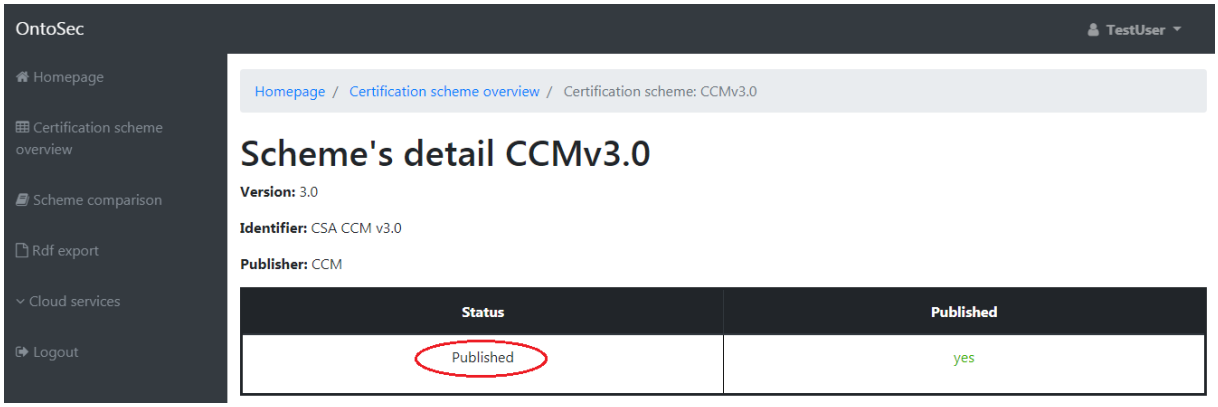
Schválenie schémy

Keď už sú všetky časti schémy schválené, celá schéma sa dostáva do stavu **Approved** (Schválená) a na jej dashboard-e sa objaví tlačidlo pre jej publikovanie, ktoré vidí vlastník tejto schémy (Obr. 7).



Obrázok 7. Schválená schéma.

Po kliknutí na toto tlačidlo sa stav schémy zmení na publikovanú a už viac nemôže byť upravovaná (Obr. 8).

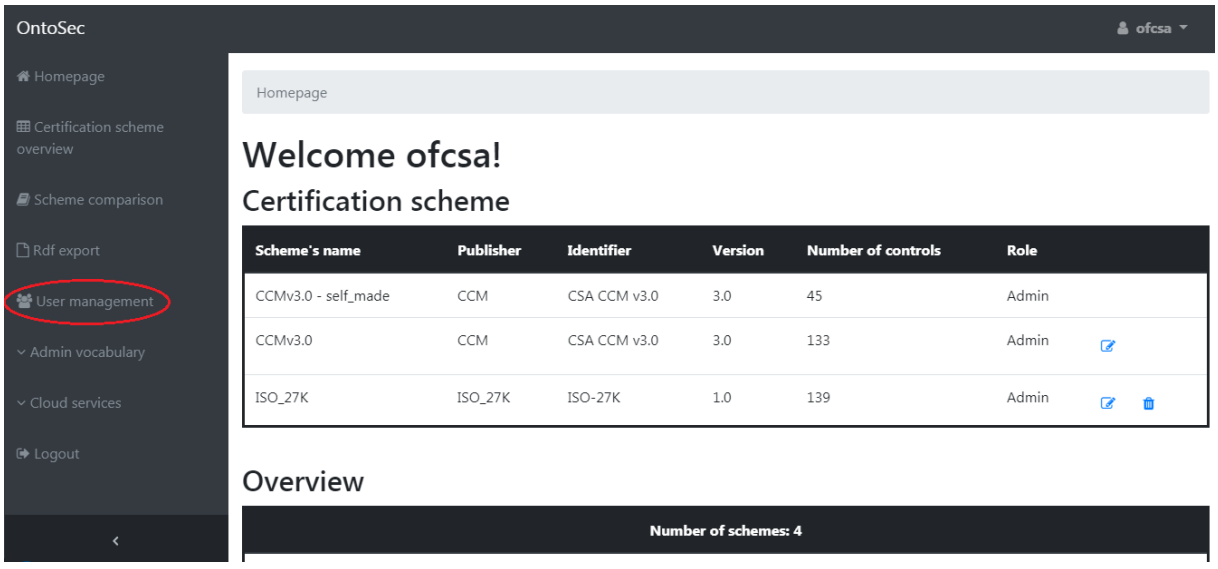


Obrázok 8. Publikovaná schéma.

Správa používateľov

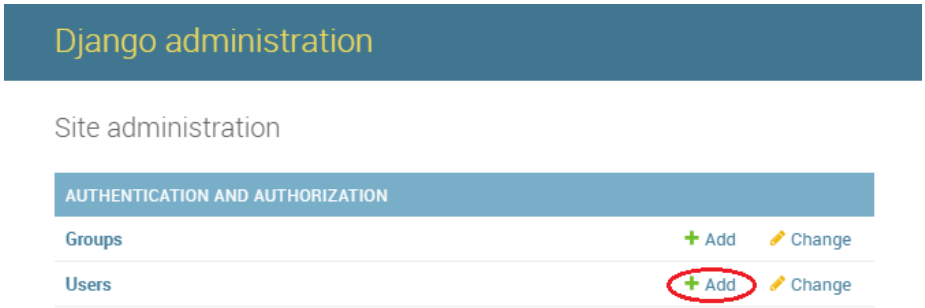
Pridanie nového používateľa

Pridanie nového používateľa sa nachádza v menu iba prihláseným používateľom a slúži na pridanie používateľa. Používatelia môžu pridávať iných používateľov, iba ak majú nastavené správcovské práva. Používateľ nastavuje práva podľa svojho uváženia a je obmedzený svojimi právami (nemôže prideliť viac práv, ako má on sám).



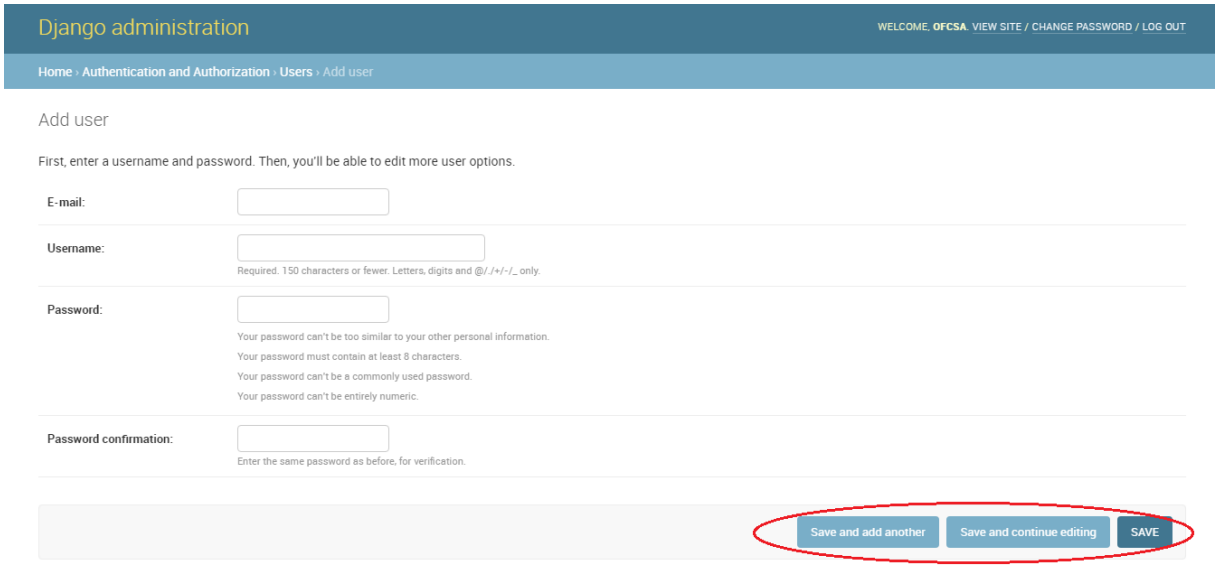
Obrázok 1. Úvodná stránka.

Po stlačení tlačidla **User management** (Správa používateľov) na úvodnej stránke (Obr. 1) sa zobrazí správcovská stránka (Obr. 2).



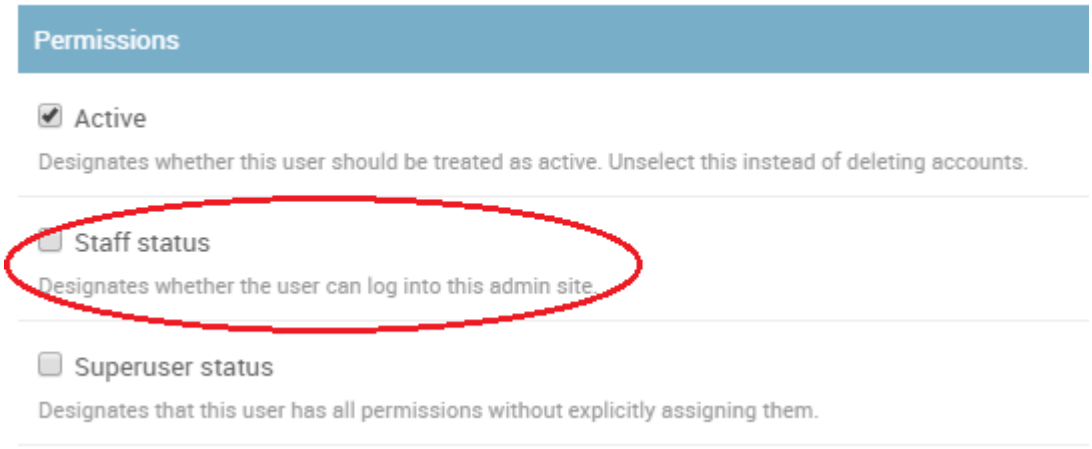
Obrázok 2. Pridanie používateľa.

Keď na tejto stránke používateľ klikne na tlačidlo **Add** (Pridať) v riadku **Users** (Používatelia) (Obr. 2), otvorí sa mu stránka s formulárom na vytvorenie nového používateľa (Obr. 3).



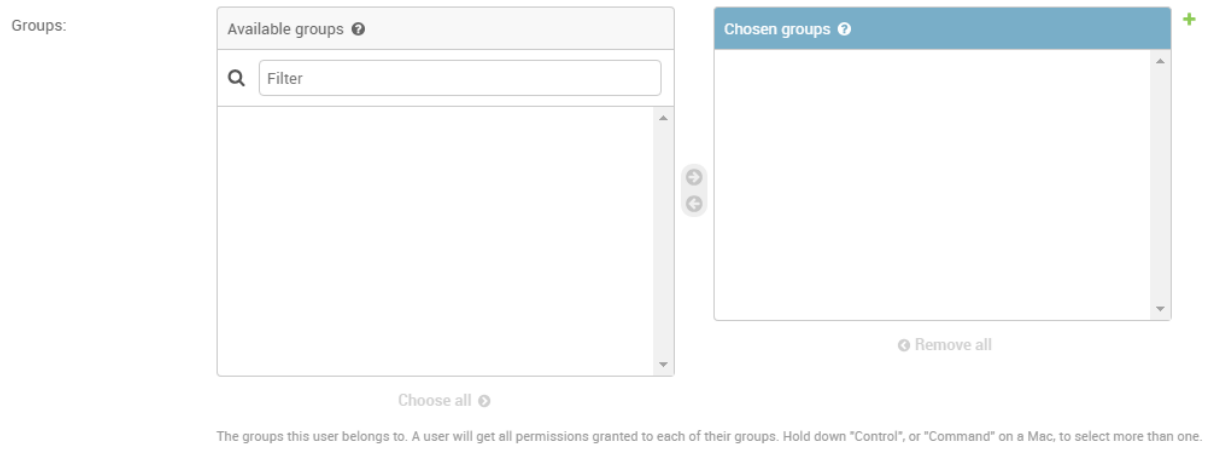
Obrázok 3. Formulár pre meno a heslo.

Vo formulári (Obr. 3) je nutné vyplniť všetky polia a následne uložiť nového používateľa, aby bolo možné neskôr mu pridávať práva.



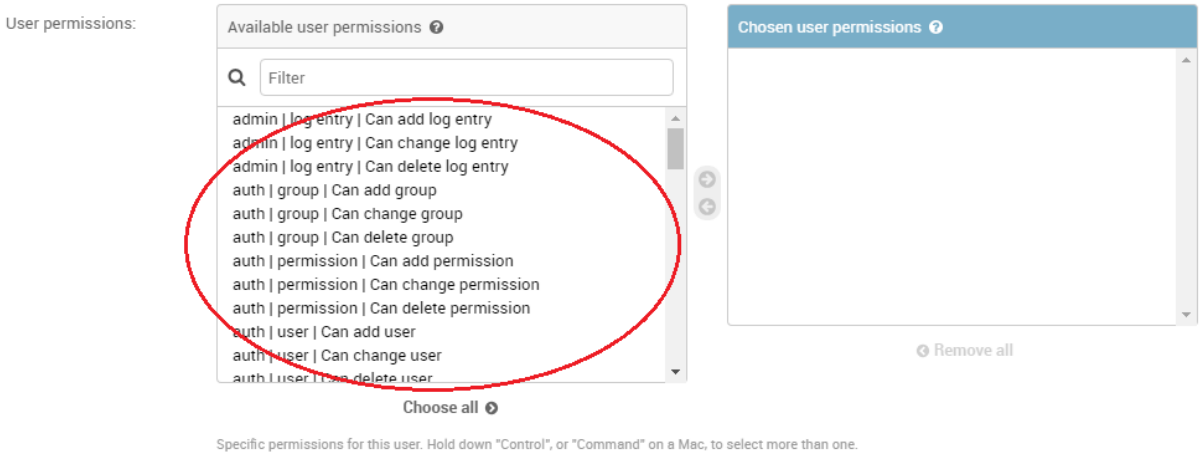
Obrázok 4. Nastavenie správcovského prístupu.

Toto políčko (Obr. 4) sa zaškrtne iba v prípade, že používateľ chce, aby nový používateľ mal možnosť pridávať nových používateľov.



Obrázok 5. Pridanie používateľa do skupiny.

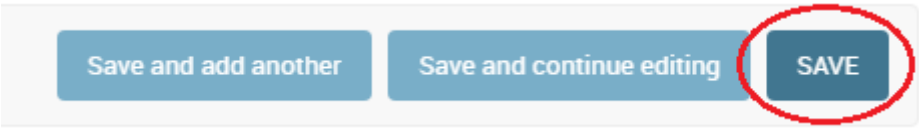
V tejto tabuľke (Obr. 5) je možné pridať vytváraného používateľa do skupiny.



Obrázok 6. Pridanie práv používateľovi.

V tejto tabuľke (Obr. 6) je možné pridať vytváranému používateľovi práva na pridanie, upravenia alebo odstránenie záznamov zo zvolených tabuliek. Povolenia sa presúvajú pomocou šípok.

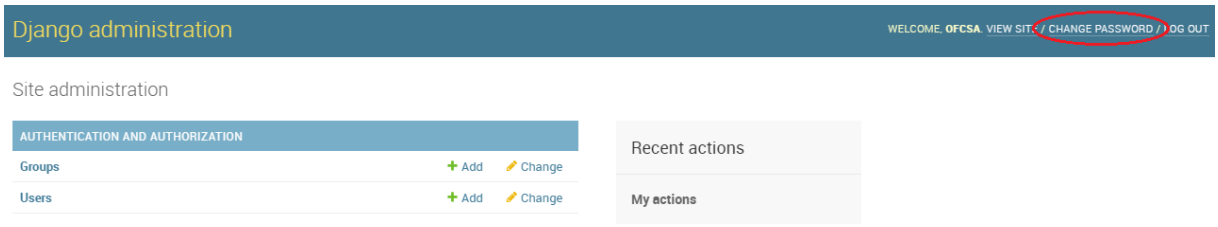
Na konci je nutné zmeny uložiť stlačením tlačidla **Save** (Uložiť) (Obr. 7), ktoré sa nachádza vpravo dole.



Obrázok 7. Tlačidlo pre uloženie používateľa.

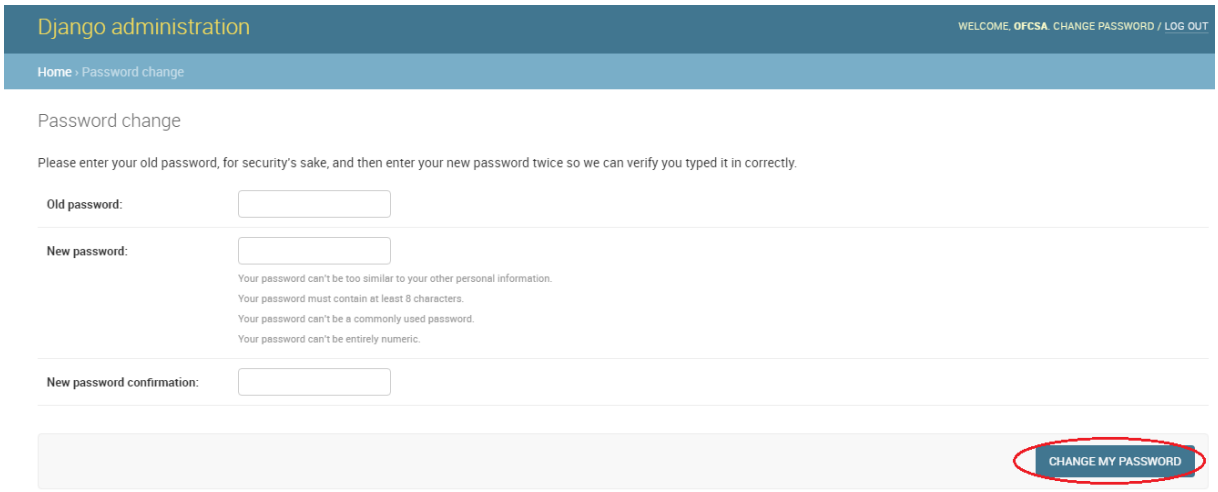
Zmena hesla

Zmena hesla používateľa sa nachádza v menu iba prihláseným používateľom a slúži na zmenu hesla (napr. vygenerovaného za vlastné). Zmena hesla sa nachádza v správe používateľov rovnako ako pridanie nového používateľa.



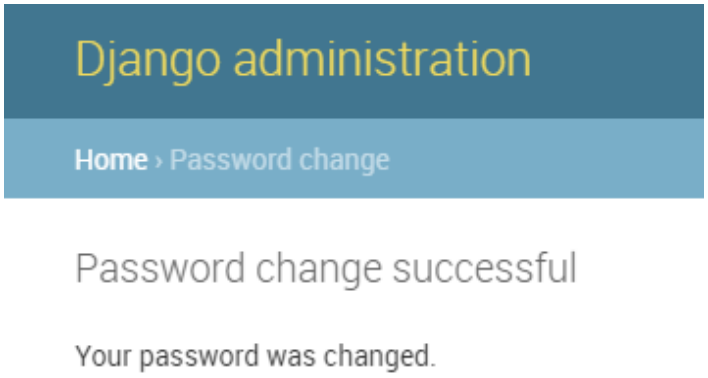
Obrázok 8. Správcovská stránka.

Na tejto stránke (Obr. 8) používateľ klikne na tlačidlo **Change password** (Zmeniť heslo) v pravom hornom rohu. Následne sa zobrazí stránka (Obr. 9) s formulárom na zmenu hesla.



Obrázok 9. Formulár na zmenu hesla.

Na tejto stránku po vyplnení starého hesla a dvakrát nového je nutné stlačiť tlačidlo **Change my password** (Zmeniť moje heslo). Ak bola zmena úspešná, zobrazí sa používateľovi informačná stránka (Obr. 10) o úspešnej zmene.



Obrázok 10. Úspešná zmena hesla.

Správa používateľských práv

V tejto časti používateľskej príručky sa nachádzajú všetky práva a role pre používateľa a návod na ich nastavovanie.

Role používateľov:

- Vlastník certifikačnej schémy
- Recenzent

Práva:

- Práva na úpravu schémy

Vlastník certifikačnej schémy

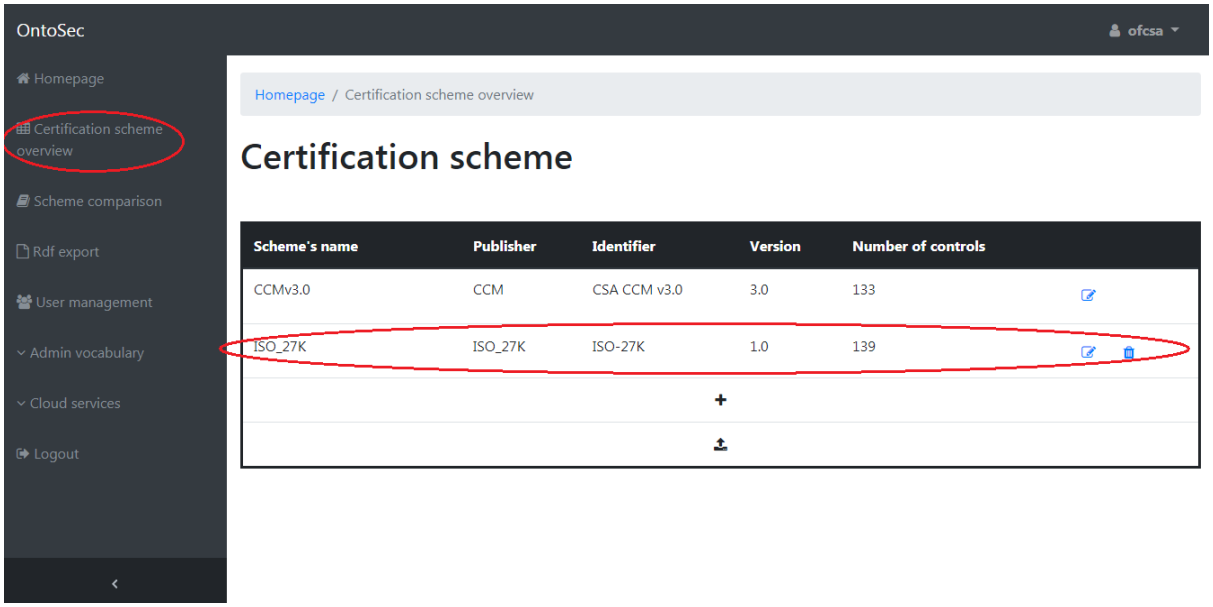
Rola vlastníka certifikačnej schémy sa nastavuje automaticky, keď používateľ vytvorí certifikačnú schému. Táto rola je pripísaná ku konkrétnej schéme a má automaticky nastavené práva na úpravu schémy. Vlastník certifikačnej schémy má možnosť schému publikovať, ak je zrecenzovaná a má schválené všetky bezpečnostné atribúty a metriky. Tlačidlo na publikovanie sa nachádza v detaile schémy (Obr. 1).

Obrázok 1. Publikovanie schémy.

Recenzent

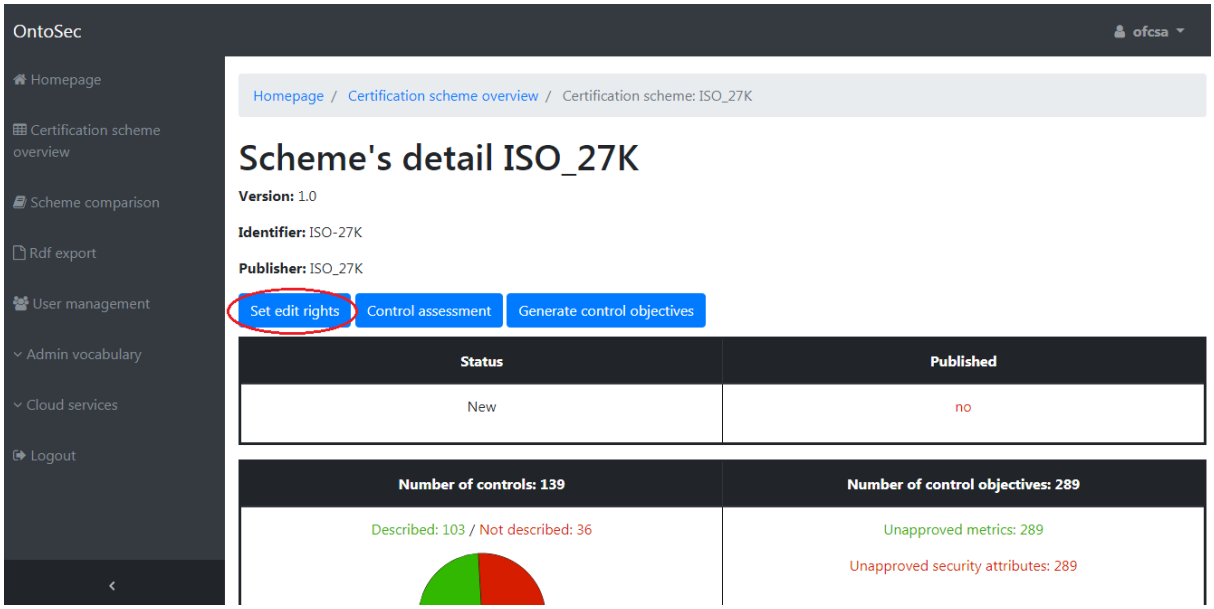
Tento používateľ môže byť hocijaký používateľ, ktorý nemá práva upravovať certifikačnú schému. Jeho úlohou je kontrola vytvorenej certifikačnej schémy a schvaľovanie jej control-ov, control objective-ov a otázok control-ov. Túto rolu môže priradiť len vlastník certifikačnej schémy.

Ako prvý krok si používateľ vyberie certifikačnú schému. Používateľ vyberie z menu ponuku pre prehľad certifikačných schém. Na stránke prehľadu certifikačných schém, môže používateľ vidieť tabuľku s certifikačnými schémami. Na vybranú certifikačnú schému používateľ klikne (Obr. 2), čo mu zobrazí detaily schémy.



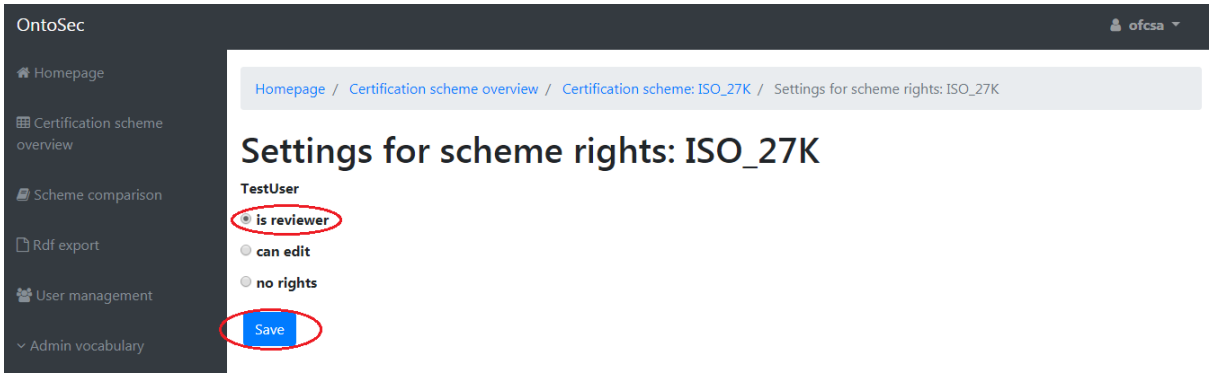
Obrázok 2. Zoznam certifikačných schém.

Na stránke detailov vybranej certifikačnej schémy používateľ klikne na tlačidlo **Set edit rights** (Nastaviť prístupové práva) (Obr. 3).



Obrázok 3. Detaily certifikačnej schémy.

Následne sa zobrazí stránka s používateľmi, ktorí majú alebo nemajú zaškrtnutú rolu recenzenta. Ak chce vlastník certifikačnej schémy niektorému používateľovi nastaviť rolu recenzenta, vyberie pod jeho menom políčko s nápisom **is reviewer** (je reviewer). Ak mu chce práva naopak odobrať, klikne na políčko s nápisom **no rights** (žiadne práva). Zmeny používateľ uloží tlačidlom **Save** (Uložiť) (Obr. 4).

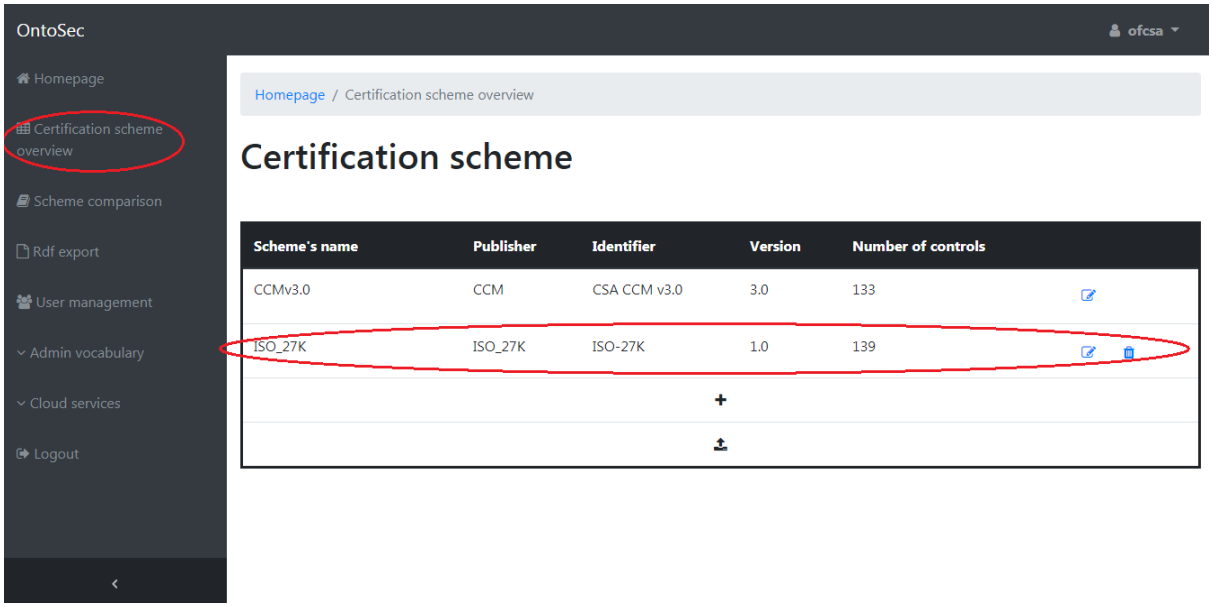


Obrázok 4. Nastavenie práv na recenzovanie certifikačnej schémy.

Používateľské práva na úpravu schémy

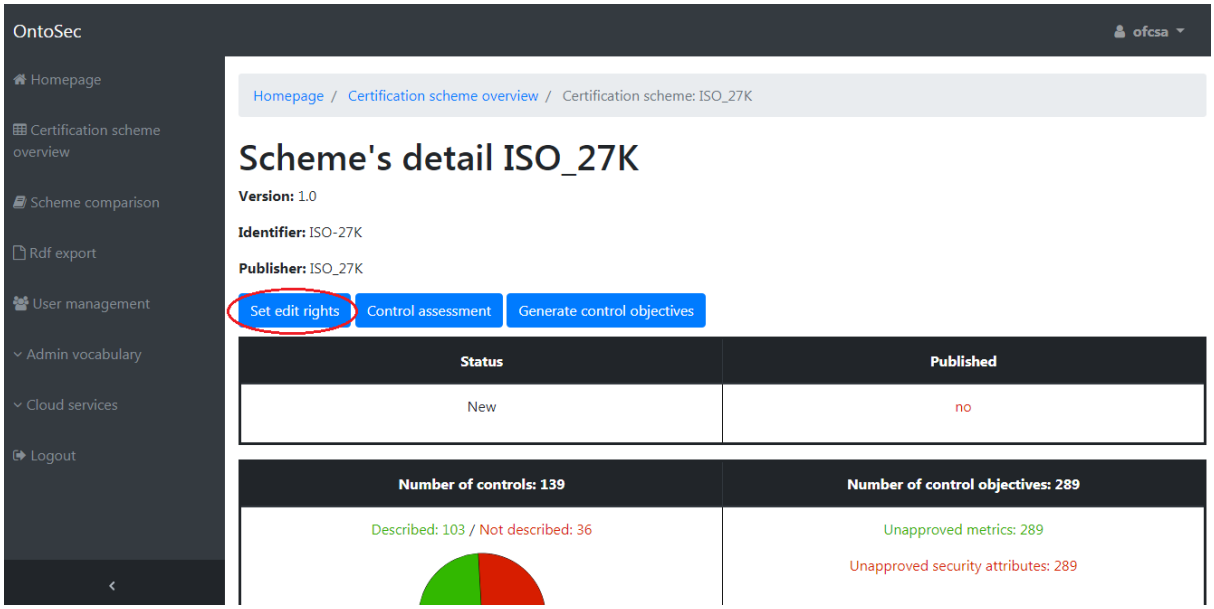
Možnosť určiť, ktorý používateľ bude môcť upravovať certifikačnú schému, môže nastaviť vlastník certifikačnej schémy po prihlásení.

Ako prvý krok si používateľ vyberie certifikačnú schému. Používateľ vyberie z menu ponuku pre prehľad certifikačných schém. Na stránke prehľadu certifikačných schém, môže používateľ vidieť tabuľku s certifikačnými schémami. Na vybranú certifikačnú schému používateľ klikne (Obr. 5), čo mu zobrazí detaily schémy.



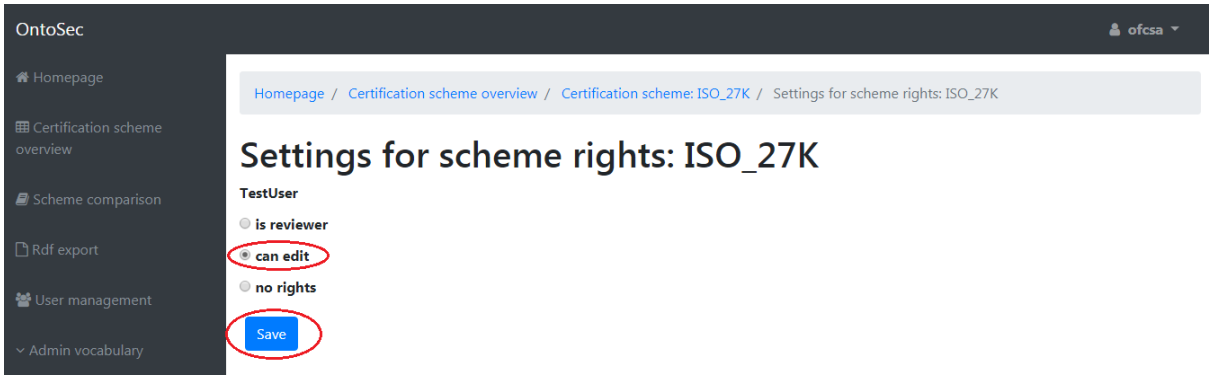
Obrázok 5. Zoznam certifikačných schém.

Na stránke detailov vybranej certifikačnej schémy používateľ klikne na tlačidlo **Set edit rights** (Nastaviť prístupové práva) (Obr. 6).



Obrázok 6. Detaily certifikačnej schémy.

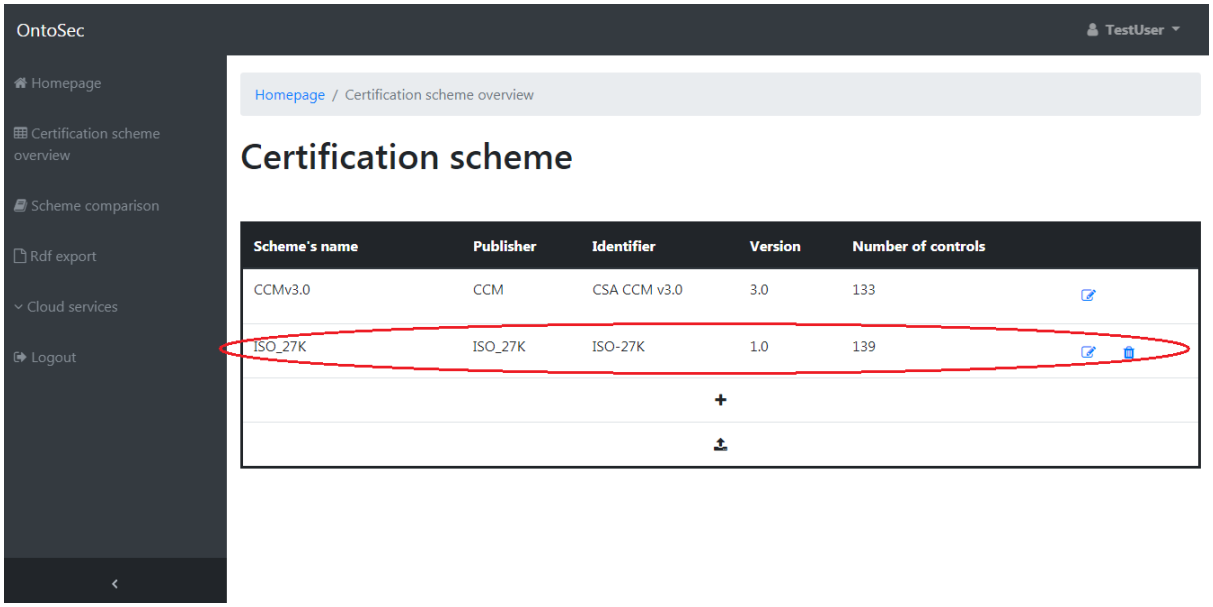
Následne sa zobrazí stránka s používateľmi, ktorí majú alebo nemajú zaškrtnuté práva na editovanie. Ak chce vlastník certifikačnej schémy niektorému používateľovi nastaviť práva na editovanie, vyberie pod jeho menom políčko s nápisom **can edit** (môže upravovať). Ak mu chce práva naopak odobrať, vyberie políčko **no rights** (žiadne práva). Zmeny používateľ uloží tlačidlom **Save** (Uložiť) (Obr. 7).



Obrázok 7. Nastavenie práv na upravovanie certifikačnej schémy.

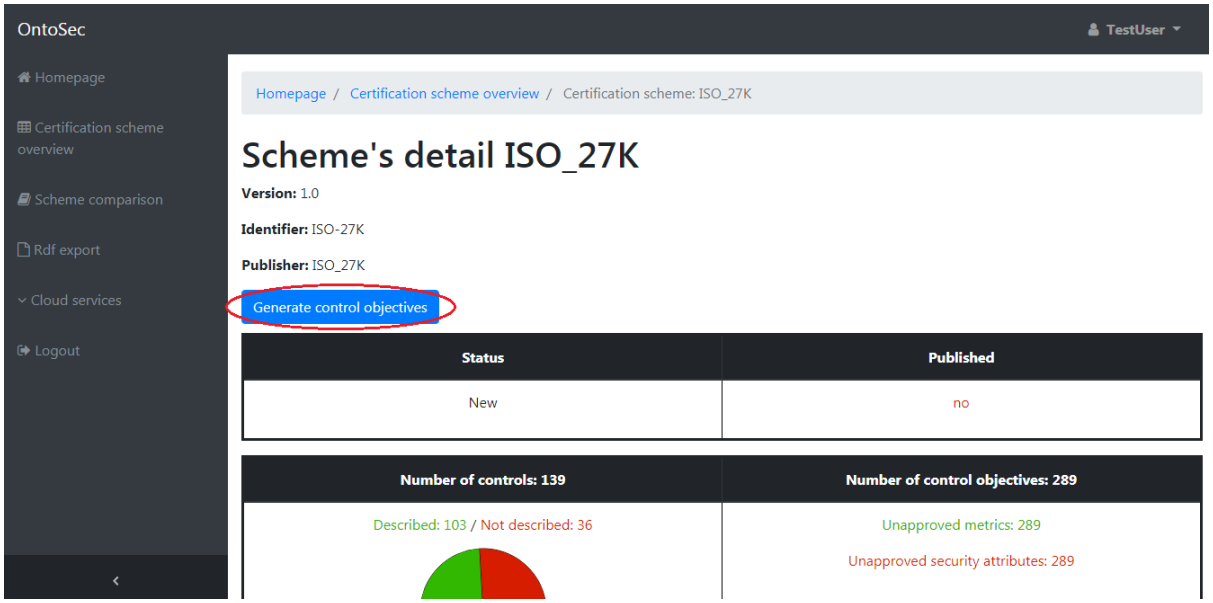
Generovanie control objective-ov

Generovanie control-ov a control objective-ov je umožnené len prihláseným používateľom. Ak používateľ chce generovať nový control objective, je potrebné, aby vybral ponuku pre prehľad certifikačných schém. Na stránke prehľadu certifikačných schém, používateľ zvolí schému, pre ktorú chce generovať control objective-y (Obr. 1).



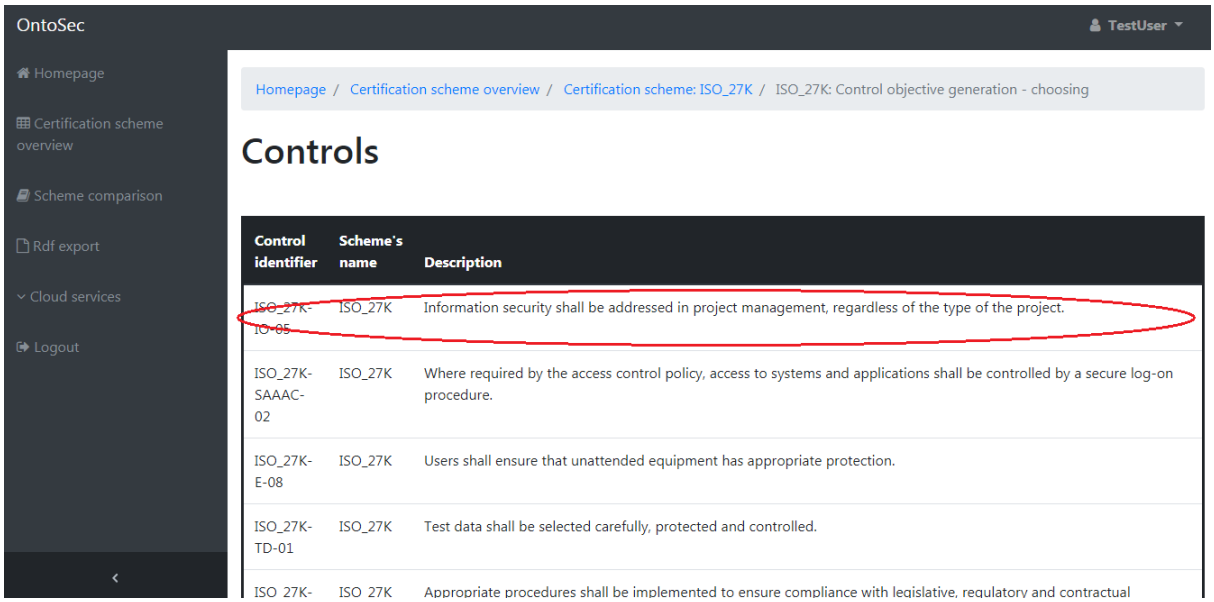
Obrázok 1. Zoznam certifikačných schém.

Používateľ následne klikne na tlačidlo **Generate control objectives** (Generovať control objective-y) (Obr. 2).



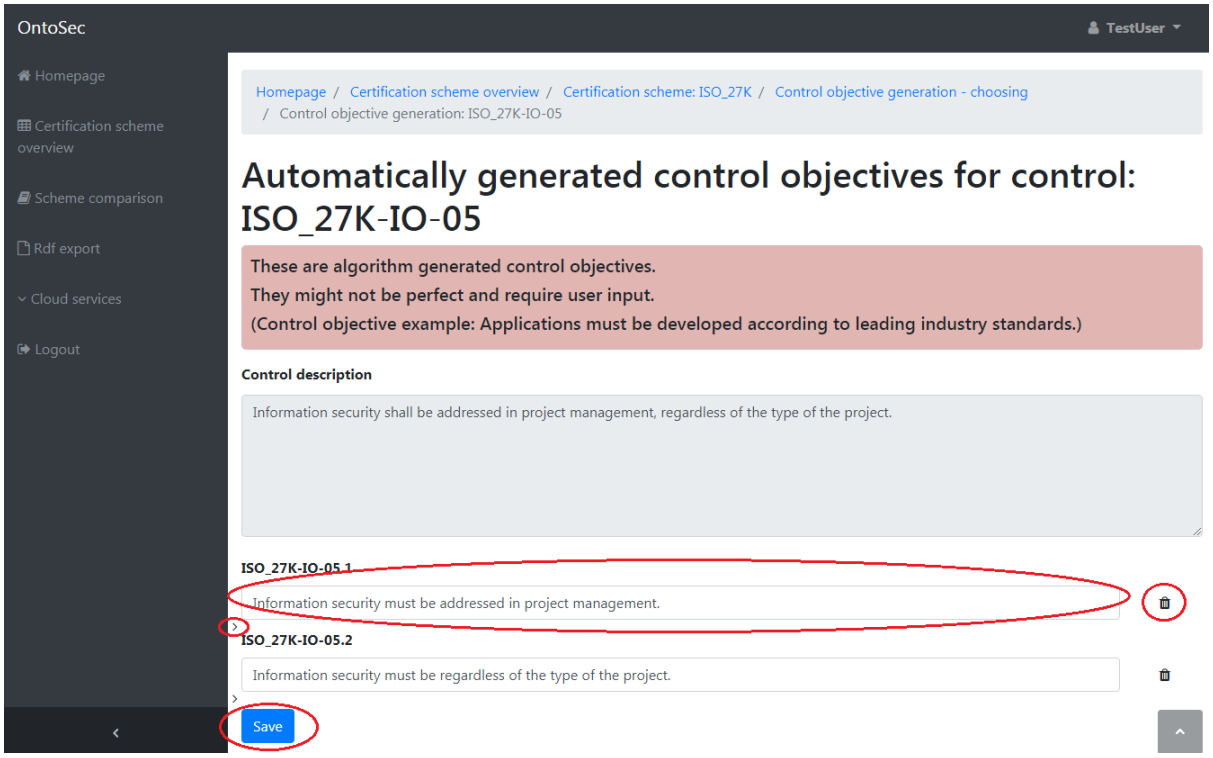
Obrázok 2. Detail schémy.

Následne sa mu zobrazí okno s control-mi, kde vyberie jeden, pre ktorý ešte neexistujú control objective-y a chce ich vytvoriť (Obr. 3).



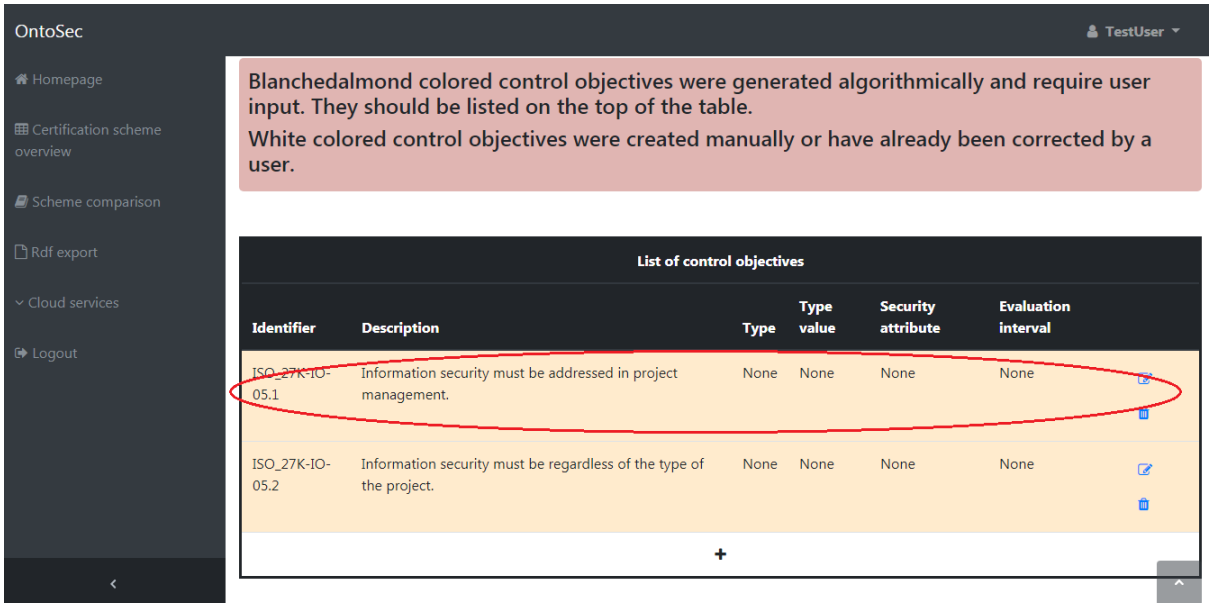
Obrázok 3. Zoznam control-ov.

Používateľ následne vidí, aké control objective-y parser analyzoval z opisu control-u. V tomto okne môže vytvárať, mazať a upravovať generované control objective-y a nakoniec ich uložiť (Obr. 4).

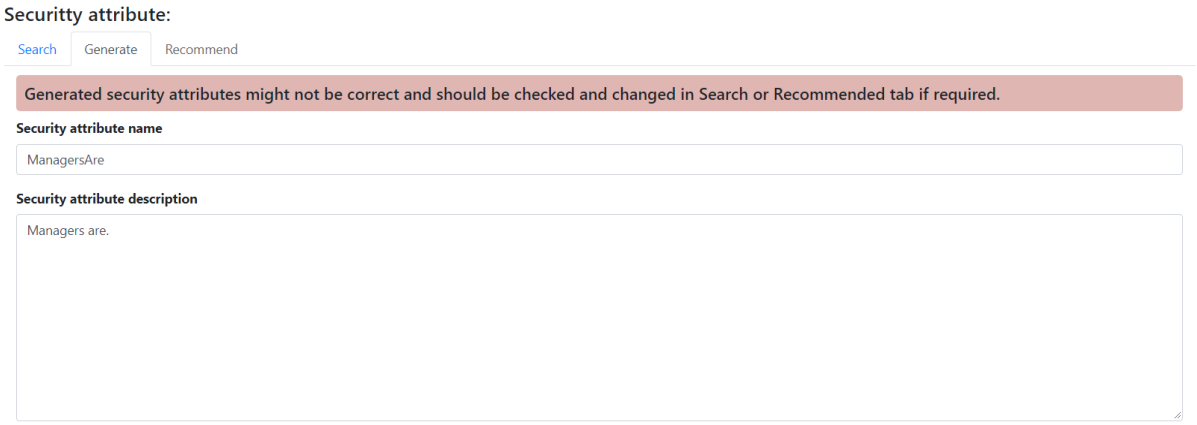


Obrázok 4. Vygenerované control objective-y pre vybraný control

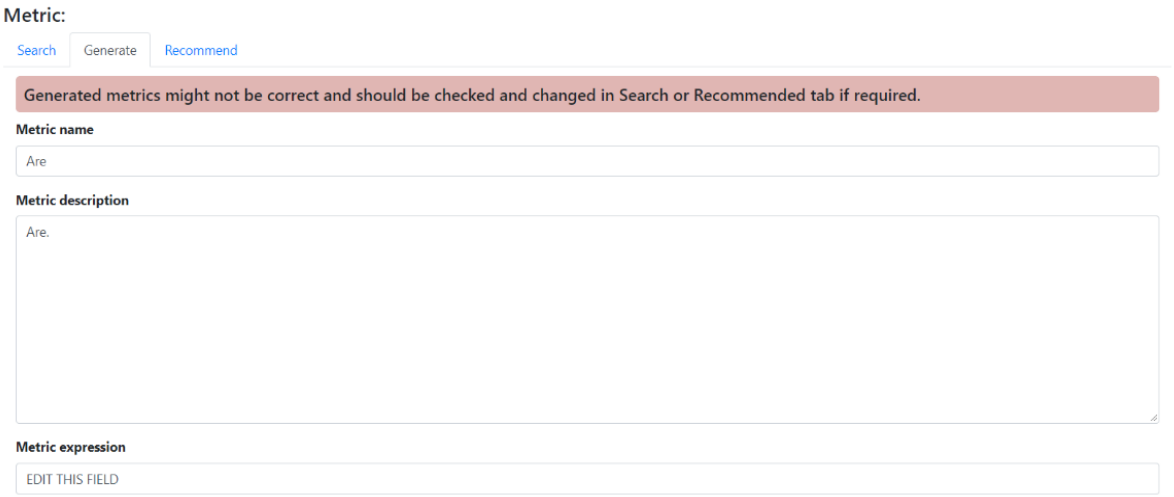
Používateľ následne vidí obrazovku, kde sú najskôr vypísané zafarbené generované control objective-y, a potom tie, ktoré už používateľ opísal (Obr. 5). Keď klikne na generovaný, otvorí sa mu okno pre jeho opis. Tu vyplní bezpečnostný atribút a metriku buď vyhľadaním, pomocou generovania alebo odporúčania (Obr. 6, 7, 8).



Obrázok 5. Generované control objective-y.



Obrázok 6. Vytváranie bezpečnostného atribútu.



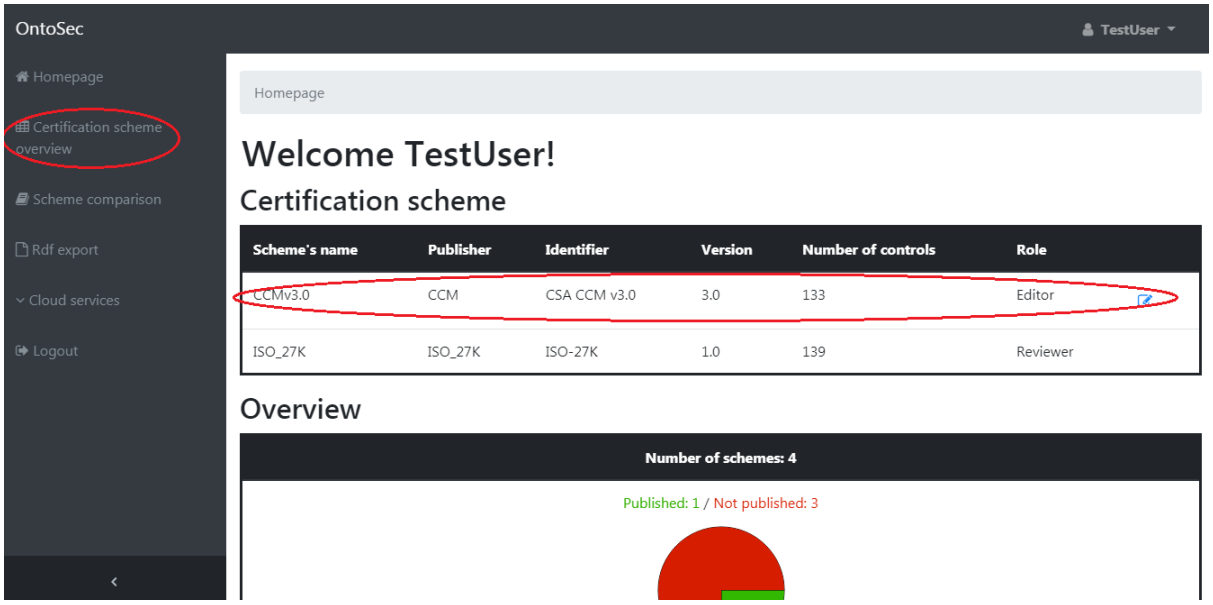
Obrázok 7. Vytváranie metriky.

Obrázok 8. Pridávanie ďalších dát control objective-u.

Následne používateľ upravený control objective uloží kliknutím na tlačidlo **Save** (Uložiť).

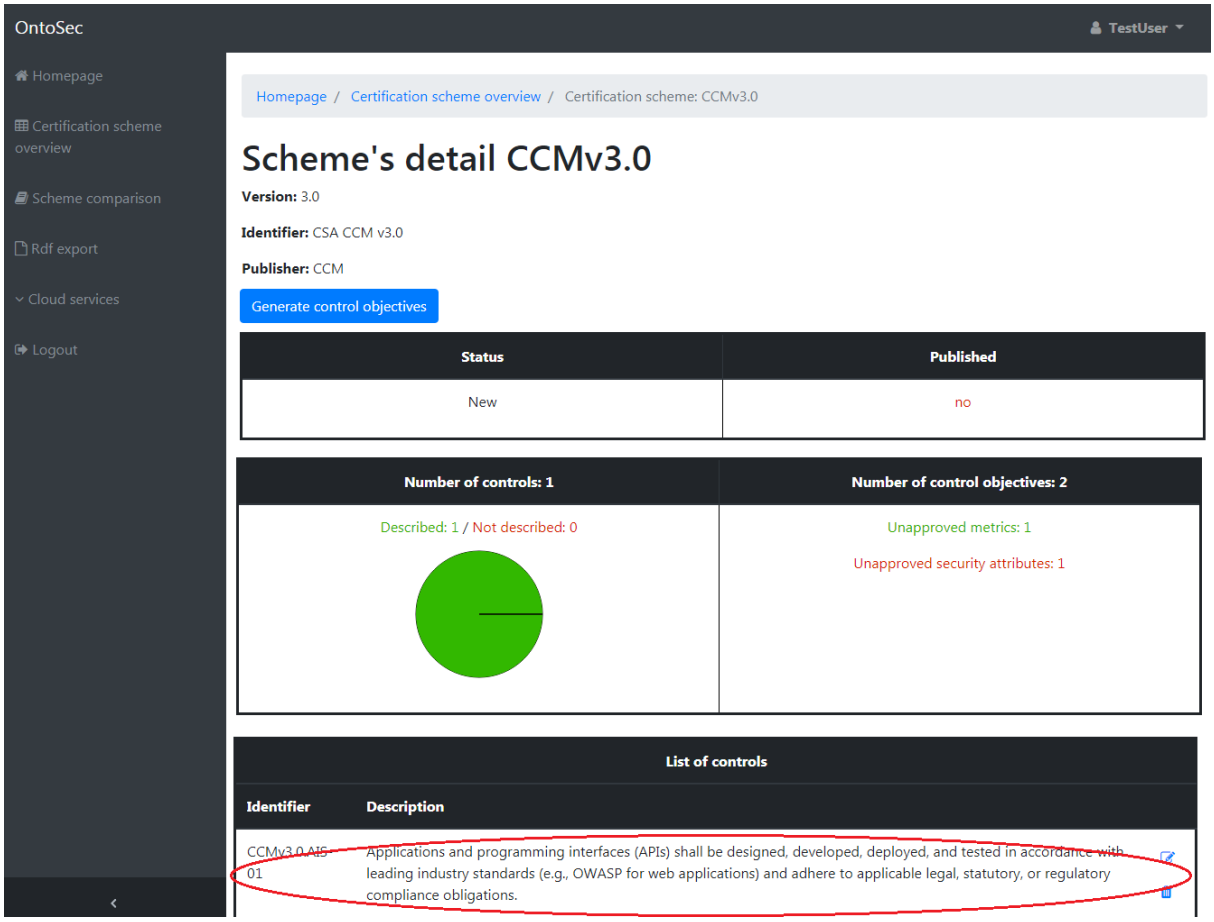
Vytvorenie nového control objective-u a otázky control-u

Po prihlásení používateľ vyberie možnosť **Certification scheme overview** (Prehľad certifikačných schém) alebo klikne na schému, ktorú môže upravovať, na svojej domovskej stránke (Obr. 1).



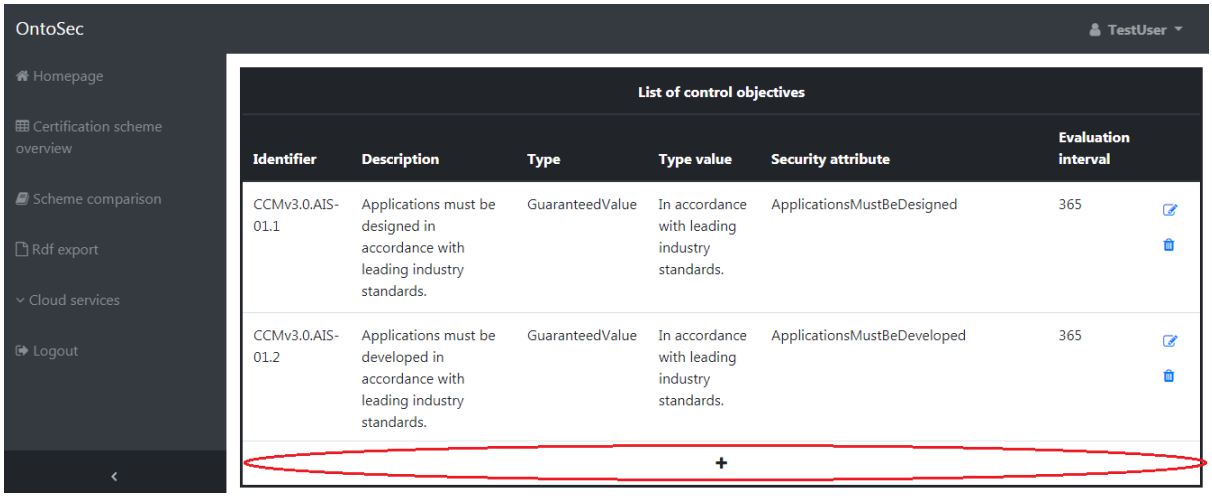
Obrázok 1. Domovská stránka.

Vo vybranej schéme vyberie konkrétny control, pre ktorý chce vytvoriť nový control objective alebo novú otázku control-u (Obr. 2).



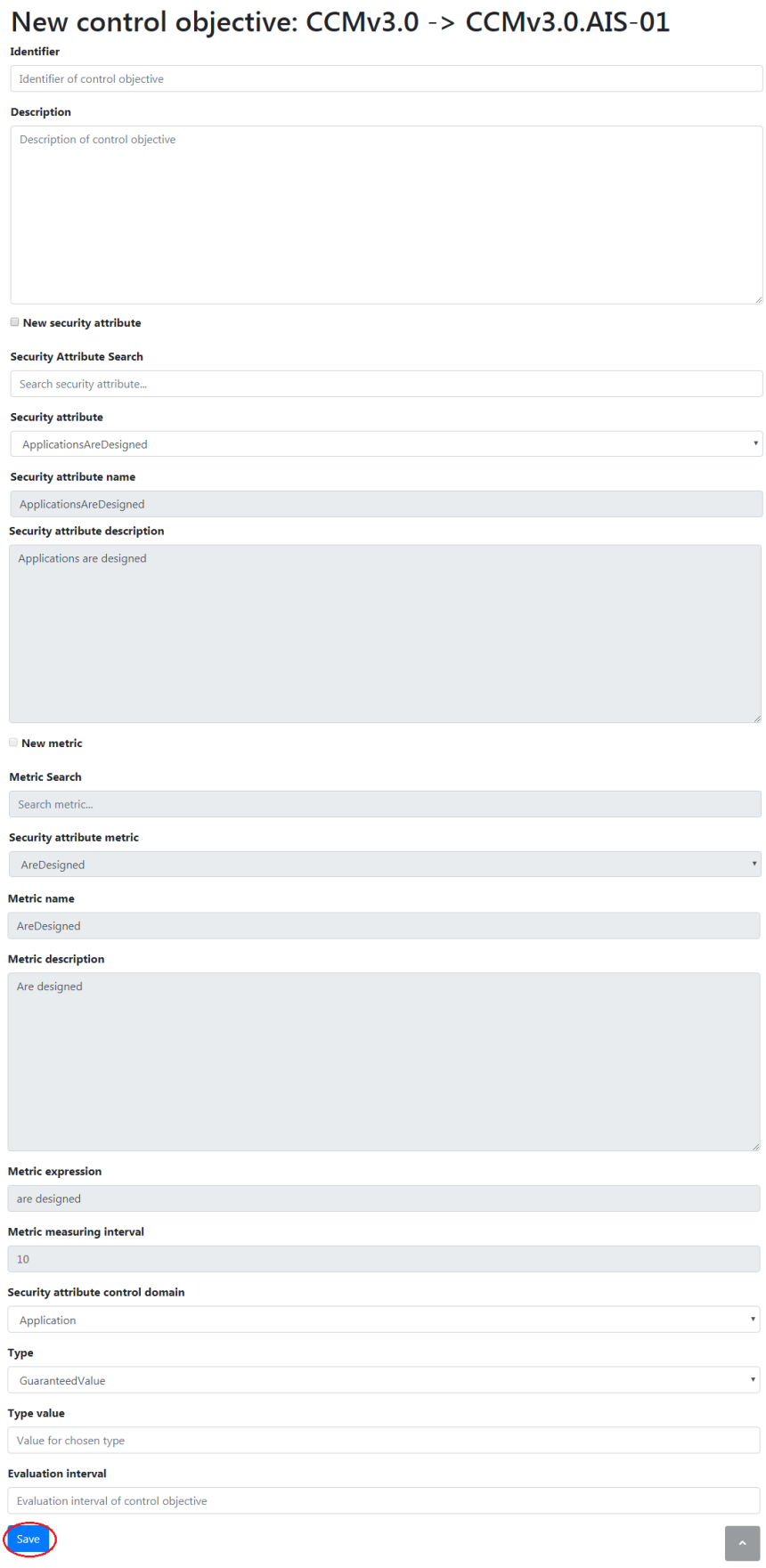
Obrázok 2. Detail schémy.

Pre vytvorenie nového control objective-u je ďalším krokom vybranie znamienka + v tabuľke so zoznamom control objective-ov (Obr. 3).



Obrázok 3. Zoznam control objective-ov.

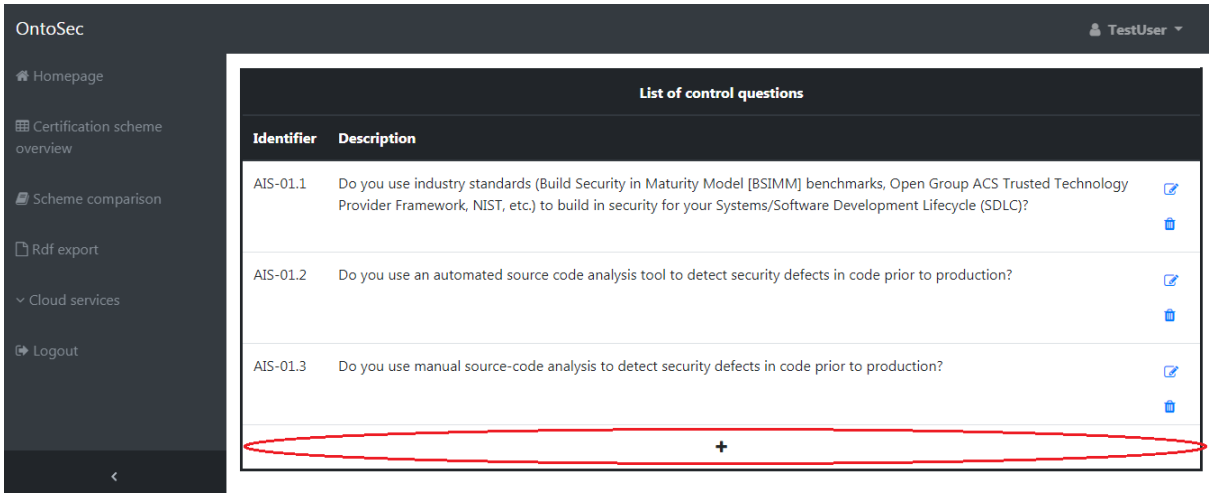
Následne sa používateľovi zobrazí stránka s formulárom, kde vyplní jednotlivé vstupné polia a stlačí tlačidlo **Save** (Uložiť), čím sa vytvorený control objective uloží do databázy (Obr. 4).



Obrázok 4. Formulár pre nový control objective.

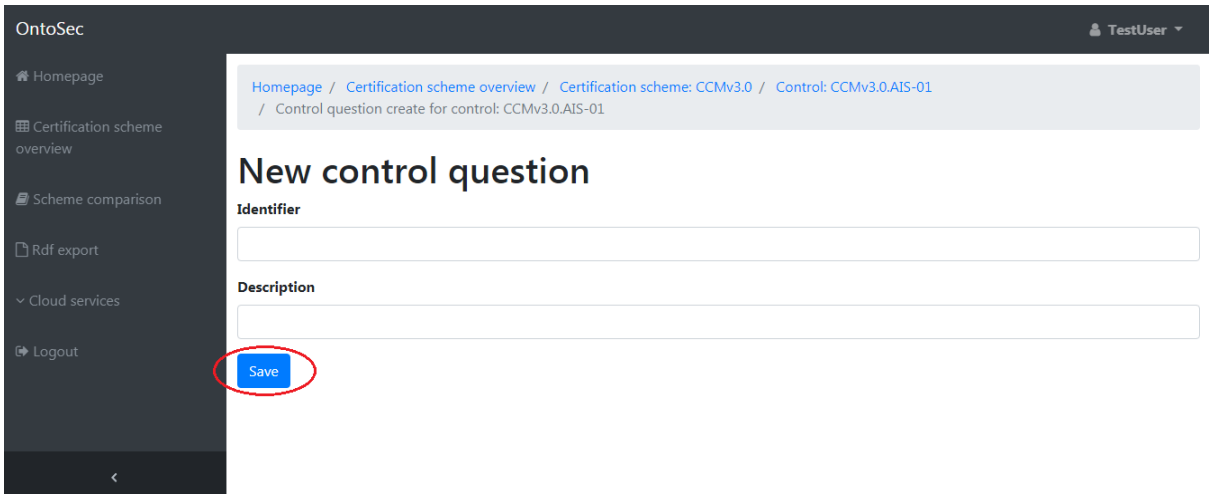
1. Identifier – unikátny identifikátor control objective-u.
2. Description – opis control objective-u.
3. Security attribute – výber existujúceho alebo vytvorenie nového atribútu.
4. Metric – výber existujúcej alebo vytvorenie novej metriky.
5. Security attribute control domain – doména bezpečnostného atribútu.
6. Type – typ control objective-u.
7. Type value – hodnota pre vybraný typ.
8. Evaluation interval – interval vyhodnocovania pre control objective.

Ak chce používateľ vytvoriť novú otázku control-u, v detaile control-u si zvolí možnosť + v tabuľke otázok control-ov (Obr. 5).



Obrázok 5. Zoznam otázok control-u.

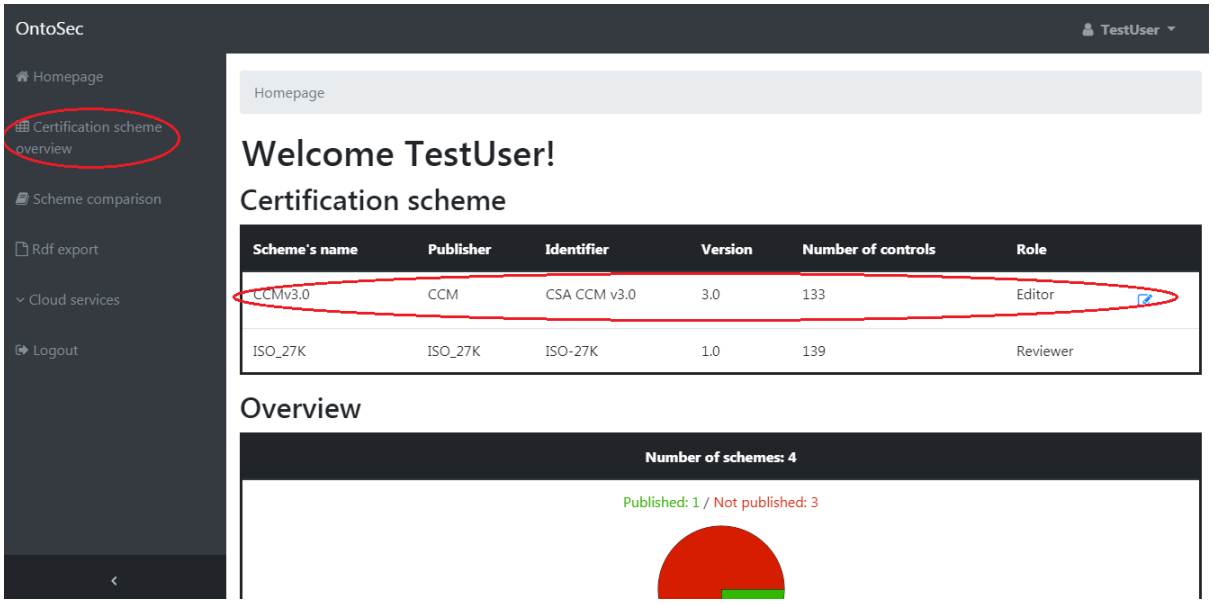
Následne sa používateľovi zobrazí stránka s formulárom, kde vyplní jednotlivé vstupné polia a otázku control-u uloží tlačidlom **Save** (Uložiť) (Obr. 6).



Obrázok 6. Vytvorenie novej otázky control-u.

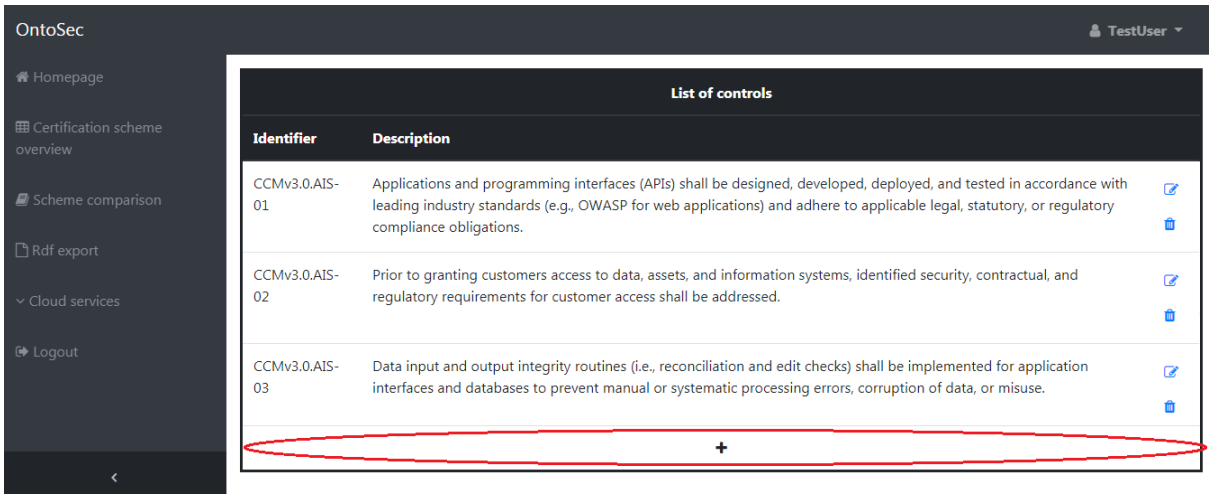
Vytvorenie nového control-u

Pridanie nového control-u je umožnené len prihláseným používateľov s vybranými právami. Pre pridanie nového control-u je potrebné, aby používateľ vybral ponuku pre prehľad certifikačných schém alebo danú certifikačnú schému vybral na svojej domovskej stránke, ktorá sa mu zobrazí po prihlásení (Obr. 1).



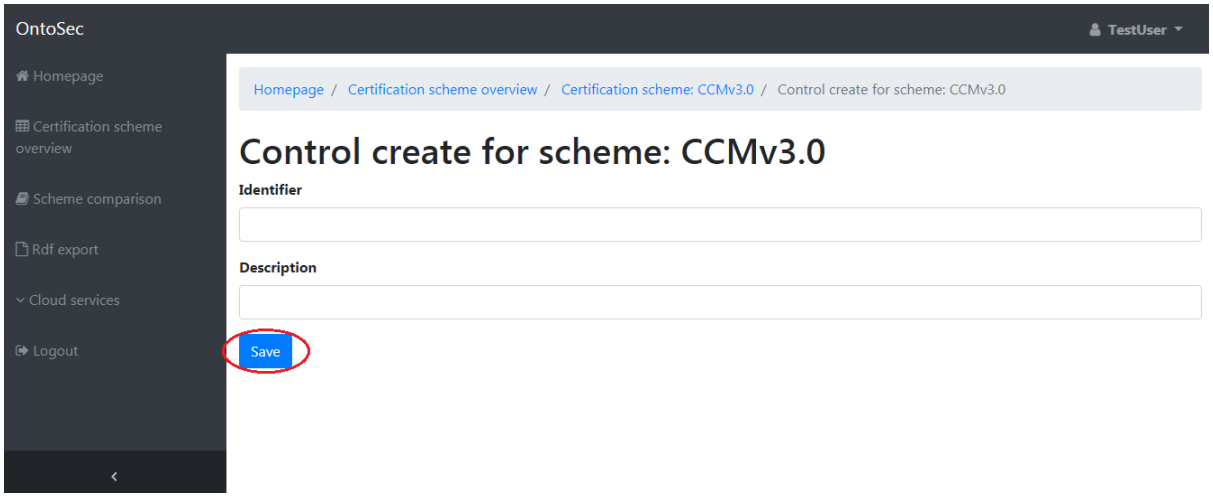
Obrázok 1. Prehľad certifikačných schém.

Na stránke vybranej certifikačnej schémy môže používateľ vidieť tabuľku so zoznamom control-ov. Kliknutím na + bude používateľ môcť vytvoriť nový control (Obr. 2).



Obrázok 2. Zoznam control-ov.

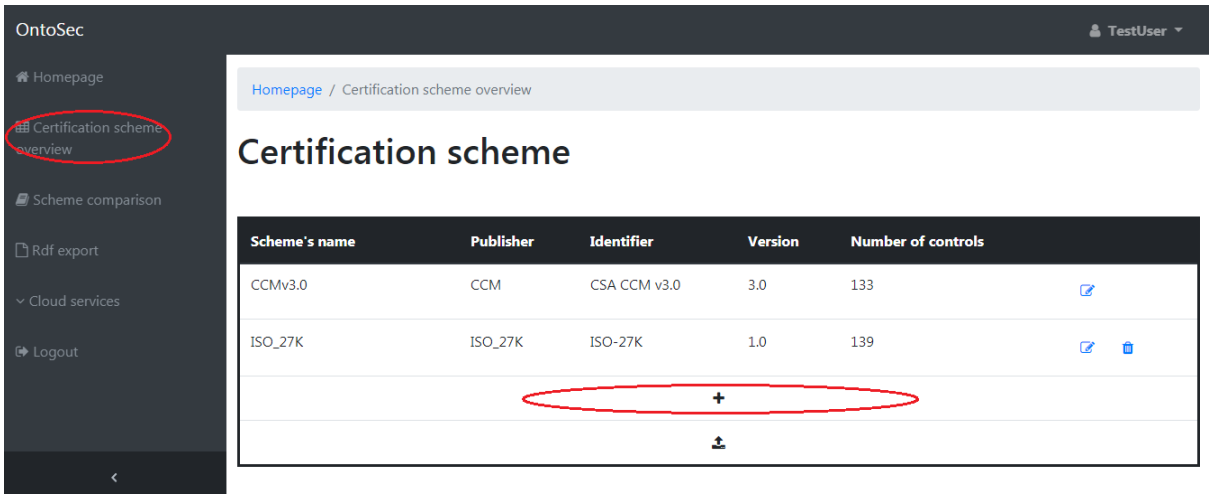
Používateľ následne vyplní pripravený formulár, kde zadá atribúty nového control-u a uloží nový control pomocou tlačidla **Save** (Uložiť) (Obr. 3).



Obrázok 3. Vytvorenie control-u.

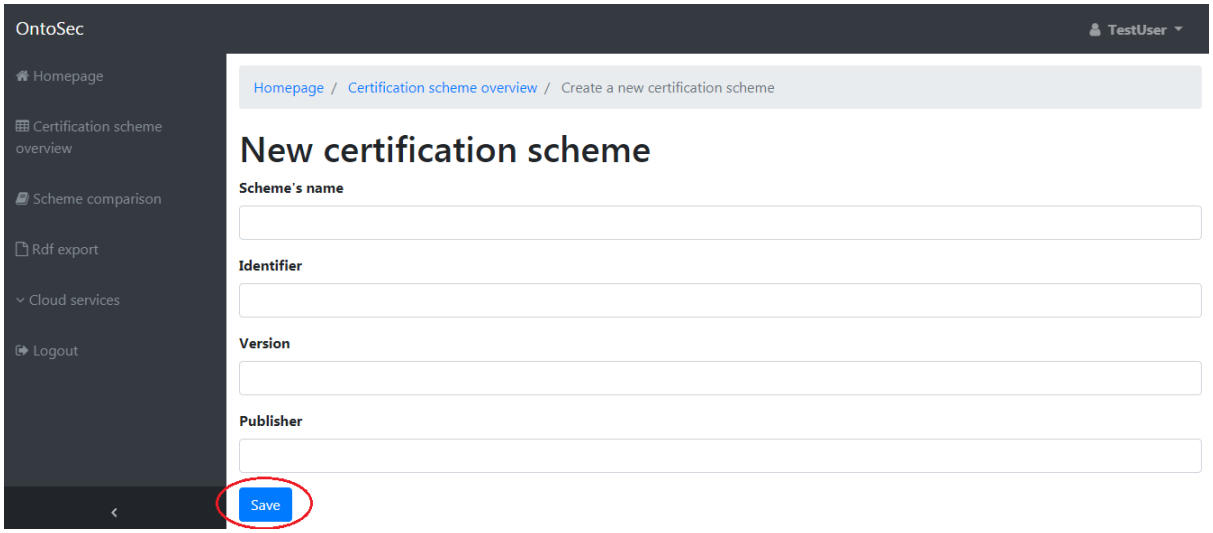
Vytvorenie novej certifikačnej schémy

Pridanie novej certifikačnej schémy je umožnené len prihláseným používateľov. Ak používateľ chce pridať novú certifikačnú schému, je potrebné, aby vybral ponuku pre prehľad certifikačných schém. Na stránke prehľadu certifikačných schém, môže používateľ vidieť tabuľku s certifikačnými schémami. Kliknutím na + je používateľ schopný vytvoriť novú certifikačnú schému (Obr. 1).



Obrázok 1. Zoznam certifikačných schém.

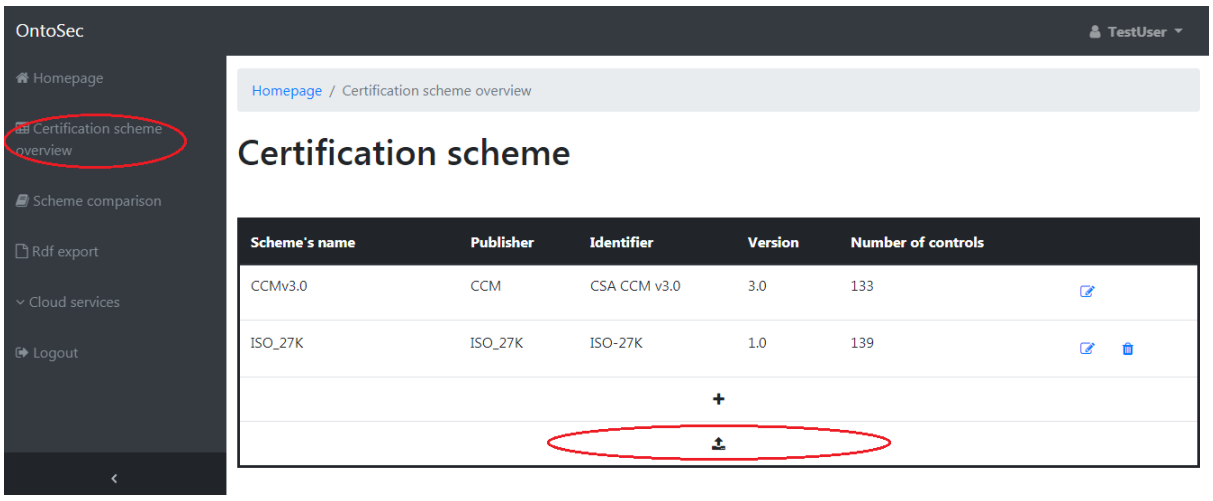
Používateľ následne vyplní pripravený formulár, kde zadá atribúty novej certifikačnej schémy. Používateľ uloží novú certifikačnú schému pomocou tlačidla **Save** (Uložiť) (Obr. 2.).



Obrázok 2. Vytvorenie certifikačnej schémy.

Importovanie novej certifikačnej schémy

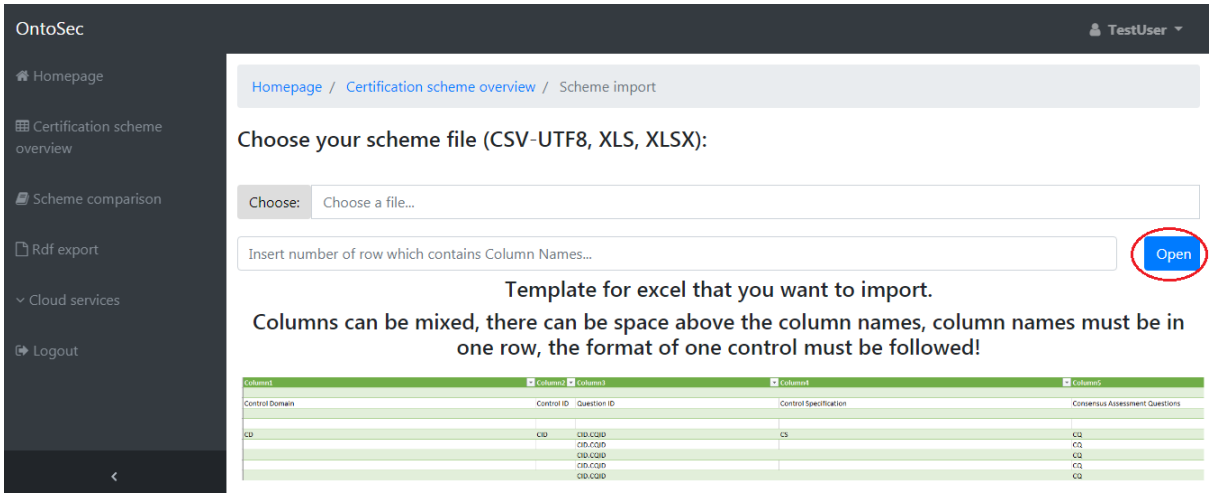
Importovanie novej certifikačnej schémy je umožnené len prihláseným používateľov. Ak používateľ chce importovať novú certifikačnú schému, je potrebné, aby vybral ponuku pre prehľad certifikačných schém. Na stránke prehľadu certifikačných schém, môže používateľ vidieť tabuľku s certifikačnými schémami. Kliknutím na ikonu pre import je používateľ schopný importovať novú certifikačnú schému (Obr. 1).



Obrázok 1. Zoznam certifikačných schém.

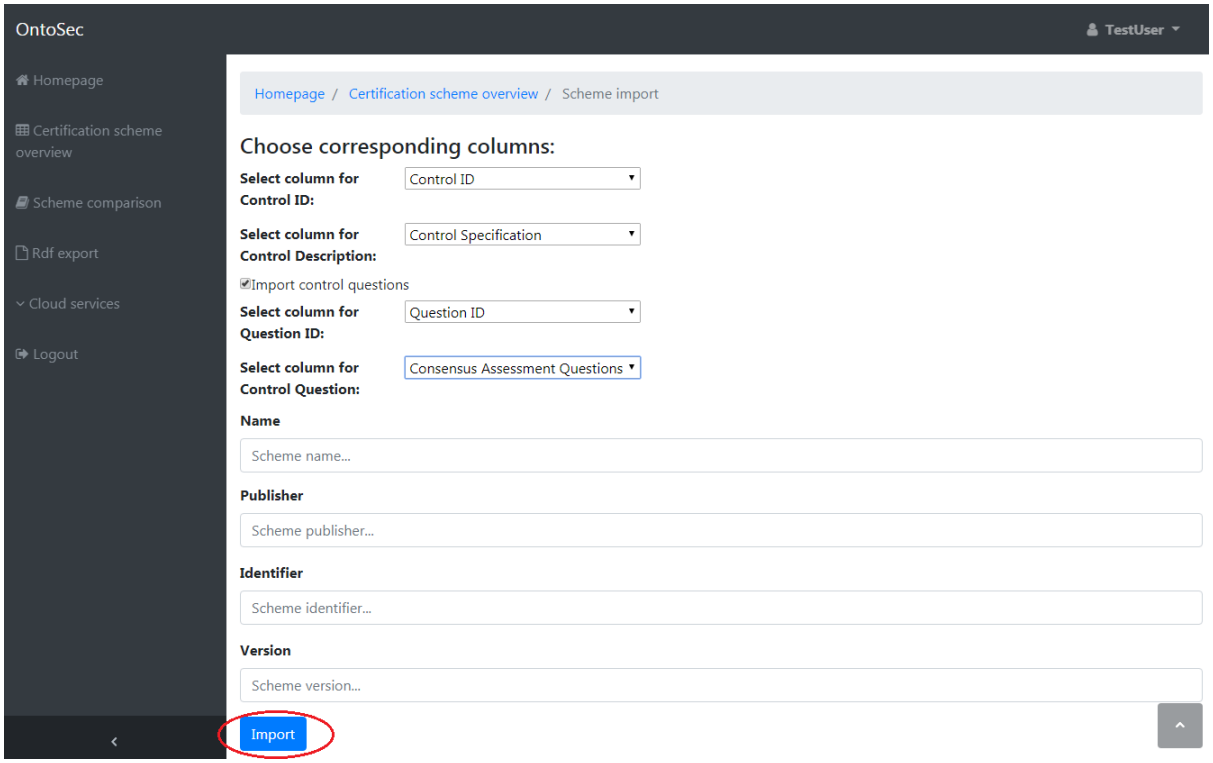
Používateľ následne vyplní pripravený formulár, kde zadá súbor na import novej certifikačnej schémy a číslo riadku, v ktorom sa nachádzajú názvy stĺpcov. Používateľovi je zobrazená aj

šablóna, ktorej sa musí držať. Používateľ otvorí novú certifikačnú schému pomocou tlačidla **Open** (Otvoriť) (Obr. 2).



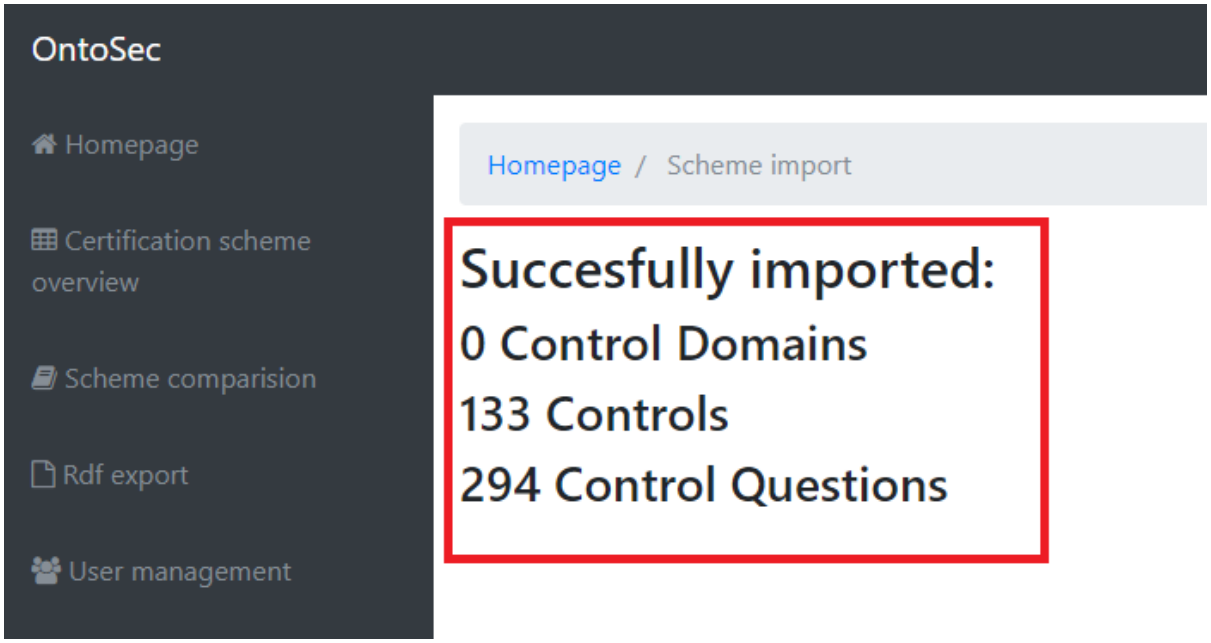
Obrázok 2. Importovanie certifikačnej schémy.

Následne sa používateľovi zobrazí formulár, kde vyplní potrebné údaje a klikne na tlačidlo **Import** (Importovať) (Obr. 3).



Obrázok 3. Importovanie certifikačnej schémy.

Používateľ následne vidí, koľko objektov sa importovalo (Obr. 4).

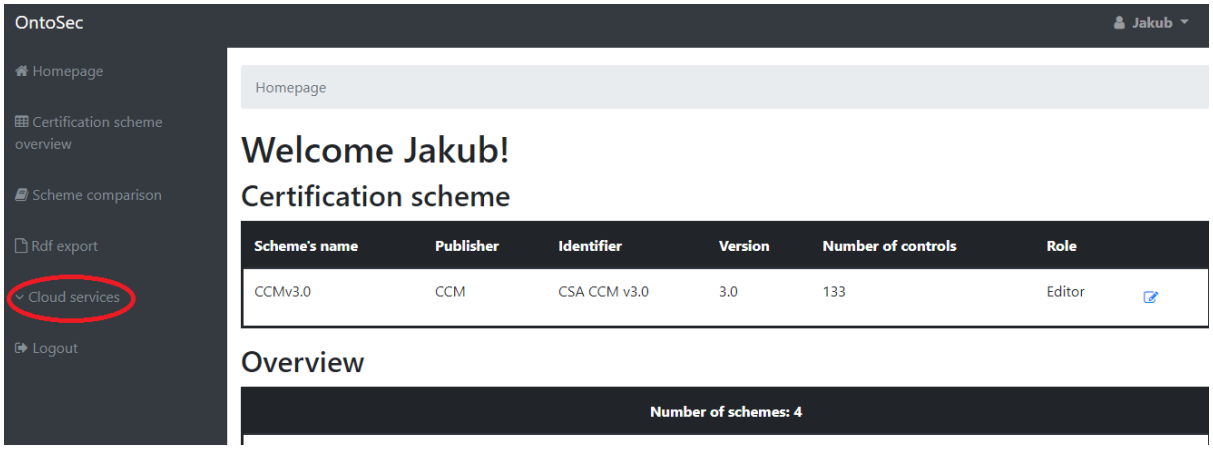


Obrázok 4. Informácie o importovanej schéme.

Správa cloudových služieb

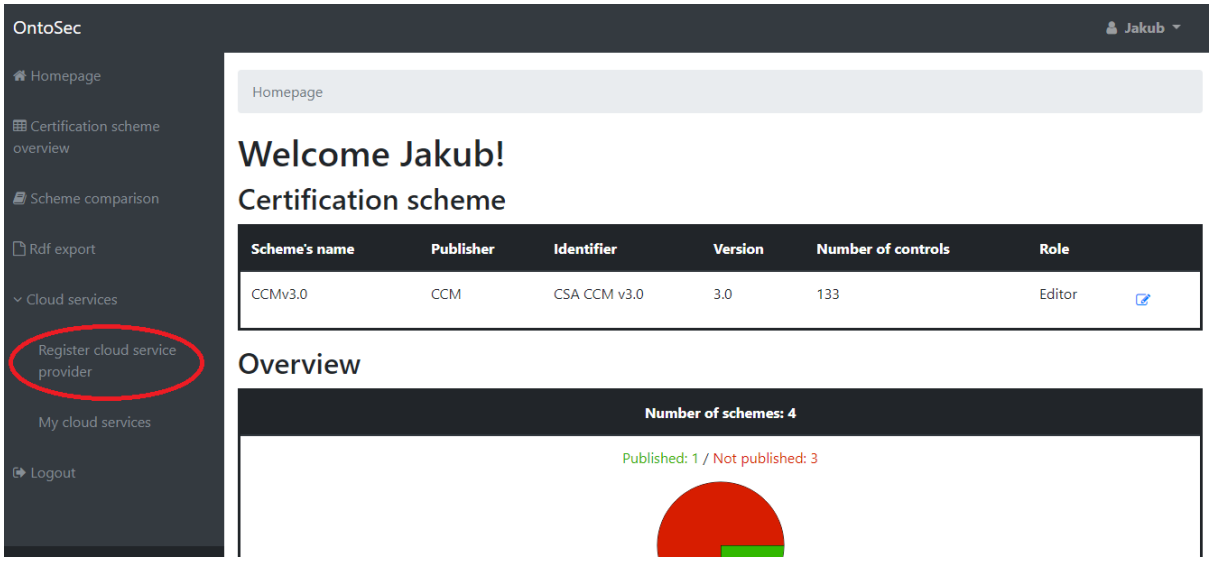
Zaregistrovanie poskytovateľa cloudových služieb

Prihlásený používateľ má v menu k dispozícii možnosť **Cloud services** (cloudové služby). Ak ešte nie je registrovaný ako poskytovateľ cloudových služieb, tak sa môže ako jeden z nich zaregistrovať.



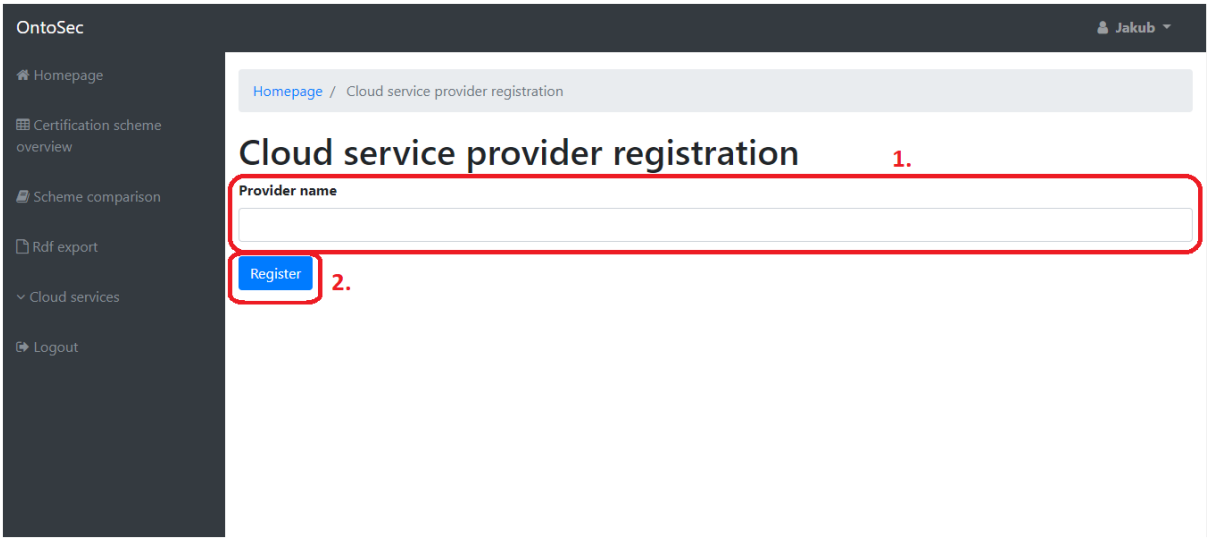
Obrázok 1. Úvodná stránka.

Po zvolení možnosti **Cloud services** (Cloudové služby), na úvodnej stránke (Obr. 1), sa používateľovi ponúknu dve možnosti (Obr. 2), z ktorých používateľ zvolí možnosť **Register cloud service provider** (Zaregistruj poskytovateľa cloudových služieb).



Obrázok 2. Zaregistrovanie poskytovateľa cloudových služieb.

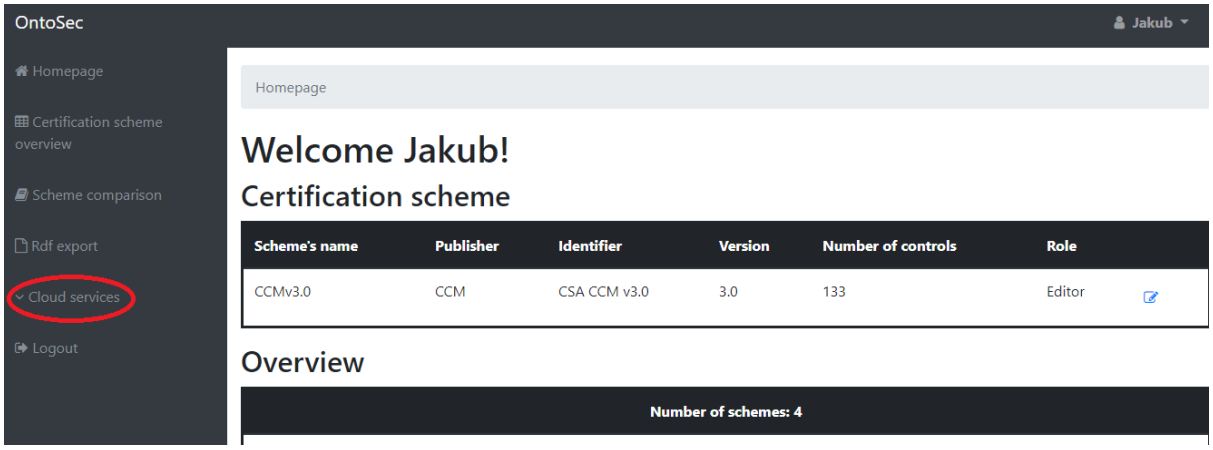
V nasledujúcom okne (Obr. 3) používateľ vyplní pole **Provider name** (Meno poskytovateľa) a klikne na tlačidlo **Register** (Registrovať sa).



Obrázok 3. Vyplnenie potrebných údajov.

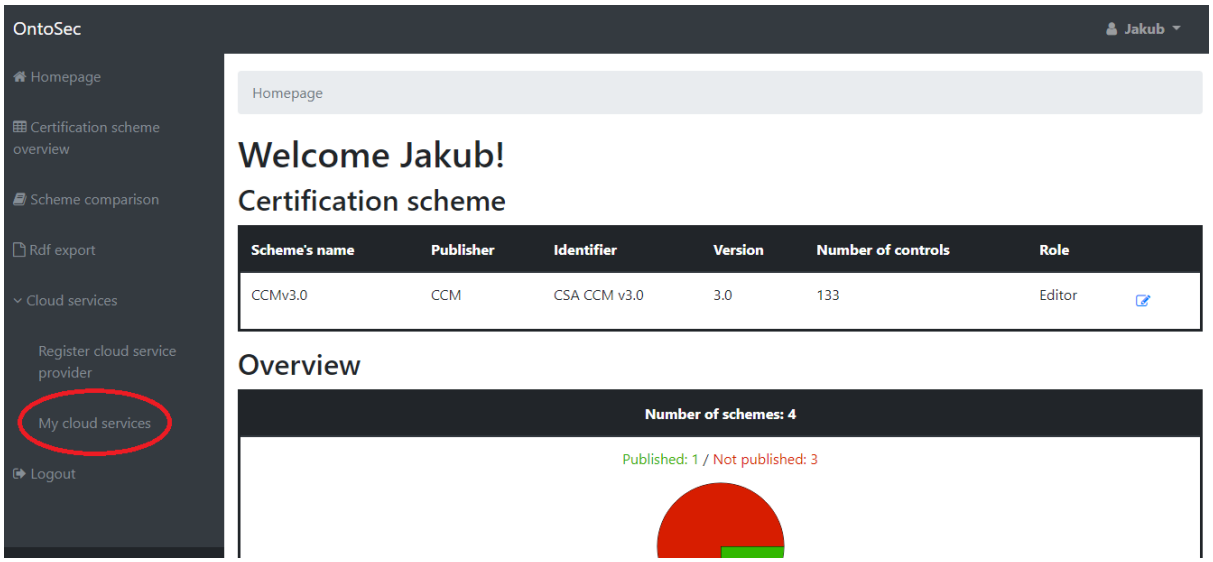
Zaregistrovanie cloudovej služby

Prihlásený používateľ má v menu k dispozícii možnosť **Cloud services** (Cloudové služby). Ak už je zaregistrovaný ako poskytovateľ cloudových služieb, tak si môže zaregistrovať cloudovú službu.



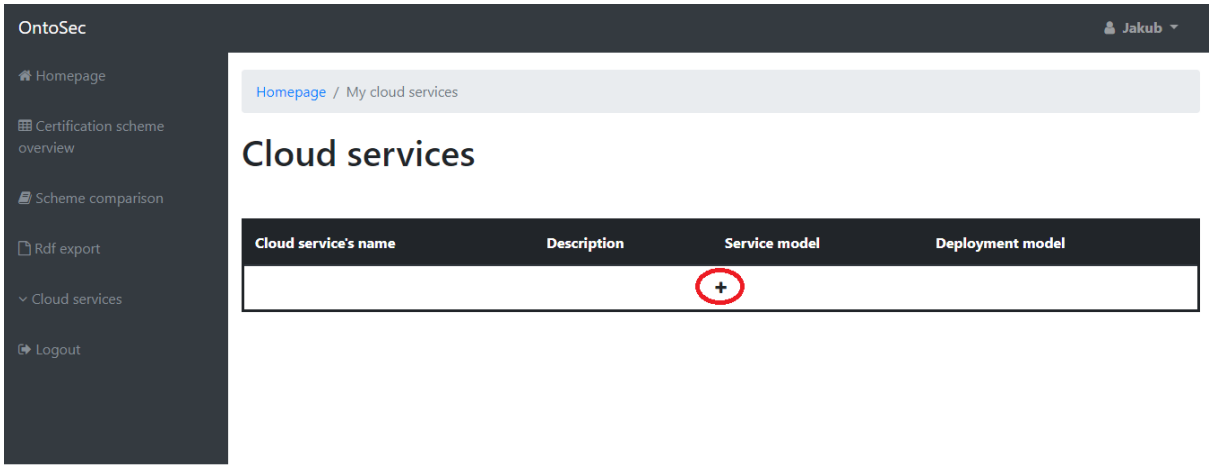
Obrázok 4. Úvodná stránka.

Po zvolení možnosti **Cloud services** (Cloudové služby), na úvodnej stránke (Obr. 4), sa používateľovi ponúknu dve možnosti (Obr. 5), z ktorých používateľ zvolí možnosť **My cloud services** (Moje cloudové služby).



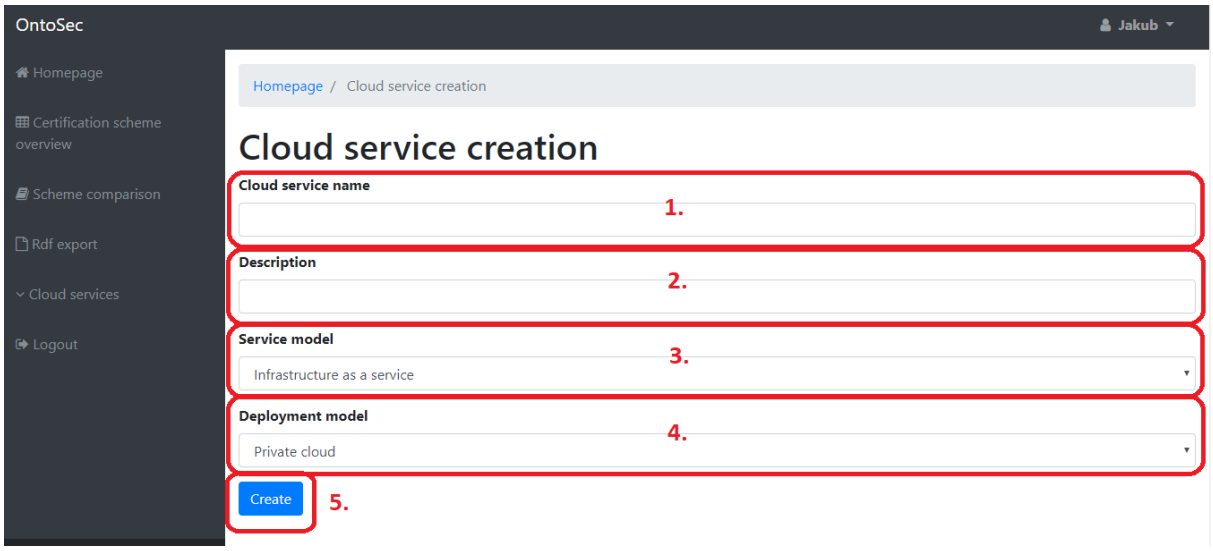
Obrázok 5. Zaregistrovanie novej cloudovej služby.

V nasledujúcom okne (Obr. 6) používateľ klikne na možnosť + reprezentujúcu pridanie novej cloudovej služby.



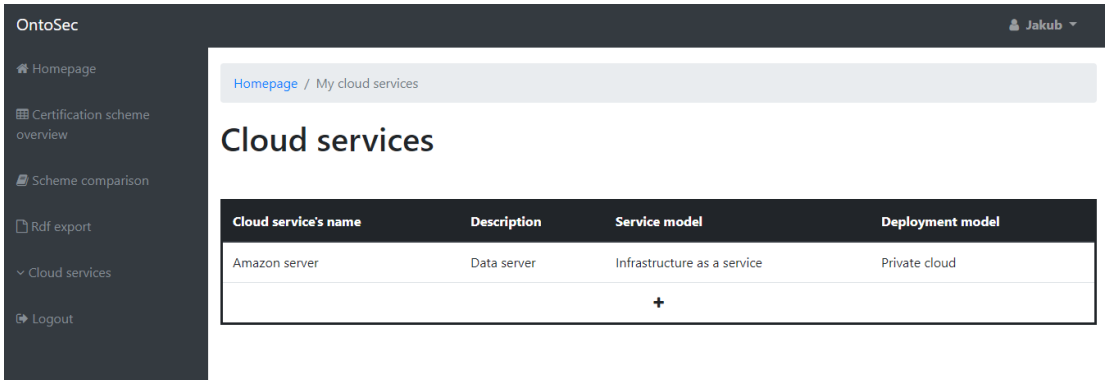
Obrázok 6. Zvolenie možnosti pridania novej služby.

Na nasledujúcej obrazovke (Obr. 7) používateľ vyplní potrebné údaje o cloudovej službe a potvrdí jej registráciu kliknutím na tlačidlo **Create** (Vytvoriť).



Obrázok 7. Vyplnenie potrebných údajov.

Po zaregistrovaní cloudovej služby je používateľ presmerovaný na prehľad svojich cloudových služieb (Obr. 8).



Obrázok 8. Prehľad cloudových služieb používateľa.

Technická dokumentácia

Vygenerovaná technická dokumentácia sa nachádza na nasledujúcich stranách.

OFCSA

1.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 cloud_service_create_form Namespace Reference

Classes

- class CloudServiceCreateForm

4.1.1 Detailed Description

```
This module holds form for cloud service creation.
```

4.2 cloud_service_provider_register_form Namespace Reference

Classes

- class CloudServiceProviderRegisterForm

4.2.1 Detailed Description

```
This module holds form for cloud service provider registration.
```

4.3 comment_form Namespace Reference

Classes

- class CommentForm

4.3.1 Detailed Description

```
This module holds form for comments.
```

4.4 control_create_form Namespace Reference

Classes

- class ControlCreateForm

4.4.1 Detailed Description

```
This module holds form for control create.
```

4.5 control_domain_select_form Namespace Reference

Classes

- class ControlDomainSelectForm

4.5.1 Detailed Description

```
This module holds form for control domain select.
```

4.6 control_objective_automatization_form Namespace Reference

Classes

- class ControlObjectiveAutomatizationForm
- class ControlObjectiveRemainsForm
- class FormSave
- class MetricFormAfterRecommend
- class MetricFormAfterSearch
- class MetricGenerateForm
- class MetricRecommendForm
- class MetricSearchForm
- class SecurityAttributeGenerateForm
- class SecurityAttributeRecommendForm
- class SecurityAttributeSearchForm

4.6.1 Detailed Description

```
This module holds form for control objective create.
```

4.7 control_objective_create_form Namespace Reference

Classes

- class ControlObjectiveCreateForm

4.7.1 Detailed Description

This module holds form for control objective create.

4.8 control_objective_edit_form Namespace Reference

Classes

- class ControlObjectiveEditForm

4.8.1 Detailed Description

This module holds form for control objective edit.

4.9 control_objective_generate_form Namespace Reference

Classes

- class ControlObjectiveGenerateForm

4.9.1 Detailed Description

This module holds form for control objective generation.

4.10 control_question_create_form Namespace Reference

Classes

- class ControlQuestionCreateForm

4.10.1 Detailed Description

This module holds form for control question create.

4.11 metric_change_form Namespace Reference

Classes

- class MetricChangeForm

4.11.1 Detailed Description

```
This module holds form for metric change.
```

4.12 my_app.admin Namespace Reference

Classes

- class UserCreationFormExtended

4.12.1 Detailed Description

```
This module holds registered the model class for admin interface
```

4.13 my_app.apps Namespace Reference

Classes

- class MyAppConfig

4.13.1 Detailed Description

```
This module contains a registry of installed applications
that stores configuration and provides introspection.
```

4.14 my_app.models Namespace Reference

Classes

- class CertificationScheme
- class CloudService
- class CloudServiceProvider
- class Control
- class ControlComment
- class ControlDomain
- class ControlObjective
- class ControlObjectiveComment
- class ControlQuestion
- class ControlQuestionComment
- class Metric
- class SchemeComment
- class SchemeRights
- class SecurityAttribute

4.14.1 Detailed Description

```
This module holds all models.
```

4.15 my_app.nlp Namespace Reference

Classes

- class NLP
- class Node

Functions

- def first_word_to_lower (root)
- def parse_stanford_tree (original_root)
- def make_word_list (sentence_nodes)
- def make_sentence (word_list)
- def traverse_tree_before_node_all (root, end_node)
- def traverse_tree_after_node_all (root, start_node)
- def traverse_tree_by_tags_all (root, tags, limited_tags=None)
- def traverse_tree_by_tags_last (root, tags, limited_tags=None)
- def traverse_tree_contains_tags (root, tags)
- def remove_successive_nodes_by_tags (nodes, tags)
- def is_active_voice (word_nodes)
- def change_voice (verb_node, to_active)
- def prepare_question_node (question_node)
- def make_paths (node)
- def prepare_traversal (root)
- def traverse_path (original_root)
- def conjugate_sentence (root)
- def prepare_next_traversal (visited_nodes)
- def traverse_tree (root)
- def get_sentences (complex_sentence)
- def get_security_attribute_from_sentence (description)
- def get_metric_from_sentence (description)
- def get_type_value_from_sentence (description)
- def recommend_security_attributes (description)
- def recommend_metrics (description)

Variables

- **config**

4.15.1 Detailed Description

```
Module for natural language processing functionality.
```

4.15.2 Function Documentation

4.15.2.1 change_voice()

```
def my_app.nlp.change_voice (
              verb_node,
              to_active )
```

Method to change the verb to it's active of passive voice with the modal verb "must".

:param verb_node: Verb node containing the verb to be changed
:param to_active: True if change to active voice, false to change to passive voice

4.15.2.2 conjugate_sentence()

```
def my_app.nlp.conjugate_sentence (
              root )
```

Method to conjugate verb in the sentence.

:param root: Root of the tree to conjugate verbs in

4.15.2.3 first_word_to_lower()

```
def my_app.nlp.first_word_to_lower (
              root )
```

If the first letter of the first word is capitalized
because of the start of the sentence, make it lowercase.

:param root: Root of the parsed sentence (leaves are words)

4.15.2.4 get_metric_from_sentence()

```
def my_app.nlp.get_metric_from_sentence (
              description )
```

Method to get metric from control objective description.

:param description: Description of the control objective
:return: Metric name and metric description

4.15.2.5 get_security_attribute_from_sentence()

```
def my_app.nlp.get_security_attribute_from_sentence (
              description )
```

Method to get security attribute from control objective description.

:param description: Description of the control objective
:return: Security attribute name and security attribute description

4.15.2.6 get_sentences()

```
def my_app.nlp.get_sentences (
              complex_sentence )
```

Method to get simple sentences from complex ones.

:param complex_sentence: Sentence to be processed
:return: List of simple sentences

4.15.2.7 get_type_value_from_sentence()

```
def my_app.nlp.get_type_value_from_sentence (
              description )
```

Method to get control objective type value from control objective description.

:param description: Description of the control objective
:return: Control objective type value

4.15.2.8 is_active_voice()

```
def my_app.nlp.is_active_voice (
              word_nodes )
```

Method to check, if the sentence is in active or passive voice.

:param word_nodes: List of word nodes, the last node should be the last verb
:return: False if passive voice, true otherwise

4.15.2.9 make_paths()

```
def my_app.nlp.make_paths (
              node )
```

Method to create paths for the node.

:param node: Node to create path for

4.15.2.10 make_sentence()

```
def my_app.nlp.make_sentence (
              word_list )
```

Method to create sentence from word_list.

:param word_list: List of words
:return: The sentence as string

4.15.2.11 make_word_list()

```
def my_app.nlp.make_word_list (
              sentence_nodes )
```

Method to create word list from sentence_nodes.

:param sentence_nodes: List of sentence nodes
:return: List of words in the sentence

4.15.2.12 parse_stanford_tree()

```
def my_app.nlp.parse_stanford_tree (
              original_root )
```

Method to parse stanford tree into out tree structure.

:param original_root: Root of the original stanford tree
:return: Root of the new tree

4.15.2.13 prepare_next_traversal()

```
def my_app.nlp.prepare_next_traversal (
              visited_nodes )
```

Method to traverse visited_nodes and try to find next traversal to take.

:param visited_nodes: List of visited nodes
:return: True if the next path was found, false otherwise.

4.15.2.14 prepare_question_node()

```
def my_app.nlp.prepare_question_node (
              question_node )
```

Method to pre-process the question.
Find the first verb node after the question node
(should contain the initial question verb)
and place after the next NP node.

:param question_node: Question node of the tree to pre-process

4.15.2.15 prepare_traversal()

```
def my_app.nlp.prepare_traversal (
              root )
```

Method to pre-process tree for the real traversal.
Creates paths for every node.

:param root: Root of the tree to prepare traversal for

4.15.2.16 recommend_metrics()

```
def my_app.nlp.recommend_metrics (
              description )
```

Method to get recommended metrics from control objective description.

:param description: Description of the control objective
:return: Data of the recommended metrics

4.15.2.17 recommend_security_attributes()

```
def my_app.nlp.recommend_security_attributes (
              description )
```

Method to get recommended security attributes from control objective description.

:param description: Description of the control objective
:return: Data of the recommended security attributes

4.15.2.18 remove_successive_nodes_by_tags()

```
def my_app.nlp.remove_successive_nodes_by_tags (
              nodes,
              tags )
```

Method to remove nodes from the tree with the specified tags.
Stops removing when the node without the specified tag is reached.

:param nodes: List of nodes to be traversed
:param tags: Tags that the node must be labeled as to be removed

4.15.2.19 traverse_path()

```
def my_app.nlp.traverse_path (
              original_root )
```

Method to traverse one path from tree.
Collect visited nodes and create new tree from the collected nodes.

:param original_root: Root of the tree to traverse
:return: Visited nodes as list, the root of the new tree

4.15.2.20 traverse_tree()

```
def my_app.nlp.traverse_tree (
              root )
```

Method to traverse tree and collect all the sentences.

:param root: Root of the tree
:return: List of sentences containing lists of words

4.15.2.21 traverse_tree_after_node_all()

```
def my_app.nlp.traverse_tree_after_node_all (
              root,
              start_node )
```

Method to traverse tree and start collecting the word nodes after the specific node.

```
:param root: Root of the tree
:param start_node: Node to indicate the start of the word nodes collection
:return: List of the collected word nodes
```

4.15.2.22 traverse_tree_before_node_all()

```
def my_app.nlp.traverse_tree_before_node_all (
              root,
              end_node )
```

Method to traverse tree and collect the word nodes before the specific node.

```
:param root: Root of the tree
:param end_node: Node to indicate the end of the word nodes collection
:return: List of the collected word nodes
```

4.15.2.23 traverse_tree_by_tags_all()

```
def my_app.nlp.traverse_tree_by_tags_all (
              root,
              tags,
              limited_tags = None )
```

Method to traverse tree by tags and collect all the word nodes.

```
:param root: Root of the tree
:param tags: List of tags that are allowed for traversal
:param limited_tags: Dictionary of the tags, that have a limit for traversal, default to None (not used)
:return: List of the collected word nodes
```

4.15.2.24 traverse_tree_by_tags_last()

```
def my_app.nlp.traverse_tree_by_tags_last (
              root,
              tags,
              limited_tags = None )
```

Method to traverse tree by tags and collect the last (right-most) word node.

```
:param root: Root of the tree
:param tags: Tags that are allowed for traversal
:param limited_tags: Dictionary of the tags, that have a limit for traversal, default to None (not used)
:return: Right-most word node of the tree that the tags allow,
None if there is no such word node
```

4.15.2.25 traverse_tree_contains_tags()

```
def my_app.nlp.traverse_tree_contains_tags (
              root,
              tags )
```

```
Method to traverse tree to find a node that contains the tags in any order.

:param root: Root of the tree
:param tags: Tags that the node must contain
:return: First node that contains the tags,
None if there is no such word node
```

4.16 my_app.rdf Namespace Reference

Functions

- def add_scheme (rdf_graph, scheme)
- def add_control (rdf_graph, control)
- def add_control_objective (rdf_graph, control_objective)
- def add_security_attribute (rdf_graph, security_attribute)
- def add_metric (rdf_graph, metric)
- def rdf_export (scheme_ids, sca_ids, metric_ids)

Variables

- **namespaces**

4.16.1 Detailed Description

```
This module holds rdf export functionality.
```

4.16.2 Function Documentation

4.16.2.1 add_control()

```
def my_app.rdf.add_control (
              rdf_graph,
              control )
```

```
Creates rdf triples for control.
```

4.16.2.2 add_control_objective()

```
def my_app.rdf.add_control_objective (
              rdf_graph,
              control_objective )
```

Creates rdf triples for control objective.

4.16.2.3 add_metric()

```
def my_app.rdf.add_metric (
              rdf_graph,
              metric )
```

Creates rdf triples for metric.

4.16.2.4 add_scheme()

```
def my_app.rdf.add_scheme (
              rdf_graph,
              scheme )
```

Creates rdf triples for scheme.

4.16.2.5 add_security_attribute()

```
def my_app.rdf.add_security_attribute (
              rdf_graph,
              security_attribute )
```

Creates rdf triples for security attribute.

4.16.2.6 rdf_export()

```
def my_app.rdf.rdf_export (
              scheme_ids,
              sca_ids,
              metric_ids )
```

This method exports certification schemes to rdf triples

4.17 my_app.search Namespace Reference

Classes

- class MetricIndex
- class SecurityAttributeIndex

Functions

- def **get_metric_index** ()
- def **get_security_atribute_index** ()
- def search (metric_name)
- def search_security (security_attribute_name)
- def search_metric_desc (description)
- def search_security_attribute_desc (description)
- def bulk_indexing ()
- def bulk_indexing_security ()

Variables

- **hosts**
- **client**
- **config**

4.17.1 Detailed Description

```
Module for elastic search functionality.
```

4.17.2 Function Documentation

4.17.2.1 bulk_indexing()

```
def my_app.search.bulk_indexing ( )
```

```
Method for bulk indexing of all metrics.
```

4.17.2.2 bulk_indexing_security()

```
def my_app.search.bulk_indexing_security ( )
```

```
Method for bulk indexing of all security attributes.
```

4.17.2.3 search()

```
def my_app.search.search (
              metric_name )
```

Method for searching metrics in elastic search.

4.17.2.4 search_metric_desc()

```
def my_app.search.search_metric_desc (
              description )
```

Method for phrase match
with metric description

4.17.2.5 search_security()

```
def my_app.search.search_security (
              security_attribute_name )
```

Method for searching security attributes in elastic search.

4.17.2.6 search_security_attribute_desc()

```
def my_app.search.search_security_attribute_desc (
              description )
```

Method for phrase match
with security attributes description

4.18 my_app.signals Namespace Reference

Functions

- def index_metric (sender, instance, kwargs)
- def remove_metric (sender, instance, kwargs)
- def index_security_attribute (sender, instance, kwargs)
- def remove_security_attribute (sender, instance, kwargs)
- def set_is_staff (sender, instance, kwargs)

Variables

- **post_save**
- **sender**

4.18.1 Detailed Description

```
This module holds all signal methods. Signal methods are run when
something defined in @receiver happens in system.
```

4.18.2 Function Documentation

4.18.2.1 index_metric()

```
def my_app.signals.index_metric (
              sender,
              instance,
              kwargs )
```

```
This method fires when metric is being saved into SQL database.
```

4.18.2.2 index_security_attribute()

```
def my_app.signals.index_security_attribute (
              sender,
              instance,
              kwargs )
```

```
This method fires when security attribute
is being saved into SQL database.
```

4.18.2.3 remove_metric()

```
def my_app.signals.remove_metric (
              sender,
              instance,
              kwargs )
```

```
This method fires when metric
is being deleted from SQL database.
```

4.18.2.4 remove_security_attribute()

```
def my_app.signals.remove_security_attribute (
              sender,
              instance,
              kwargs )
```

```
This method fires when security attribute
is being deleted from SQL database.
```

4.18.2.5 set_is_staff()

```
def my_app.signals.set_is_staff (
              sender,
              instance,
              kwargs )
```

```
This method fires when user is being saved into SQL database.
```

4.19 my_app.urls Namespace Reference

Variables

- **urlpatterns**
- string **name** = "index"),

4.19.1 Detailed Description

```
This module holds all urls.
```

4.20 my_app.views.cloud_service_create_view Namespace Reference

Classes

- class CloudServiceCreateView

4.20.1 Detailed Description

```
This module holds view for cloud service create.
```

4.21 my_app.views.cloud_service_overview_view Namespace Reference

Classes

- class CloudServicesOverviewView

4.21.1 Detailed Description

```
This module holds view for scheme overview.
```

4.22 my_app.views.cloud_service_provider_register_view Namespace Reference

Classes

- class CloudServiceProviderRegisterView

4.22.1 Detailed Description

```
This module holds view for cloud service provider registration.
```

4.23 my_app.views.control_approval_view Namespace Reference

Classes

- class ControlApprovalView

4.23.1 Detailed Description

```
This module holds view for control approval.
```

4.24 my_app.views.control_assessment_view Namespace Reference

Classes

- class ControlAssessmentView

4.24.1 Detailed Description

```
This module holds view for control assessment.
```

4.25 my_app.views.control_create_view Namespace Reference

Classes

- class ControlCreateView

4.25.1 Detailed Description

```
This module holds view for control create.
```

4.26 my_app.views.control_delete_view Namespace Reference

Classes

- class ControlDeleteView

4.26.1 Detailed Description

```
This module holds view for control delete.
```

4.27 my_app.views.control_detail_view Namespace Reference

Classes

- class ControlDetailView

4.27.1 Detailed Description

```
This module holds view for control detail.
```

4.28 my_app.views.control_edit_view Namespace Reference

Classes

- class ControlEditView

4.28.1 Detailed Description

```
This module holds view for control edit.
```

4.29 my_app.views.control_objective_approval_view Namespace Reference

Classes

- class ControlObjectiveApprovalView

4.29.1 Detailed Description

```
This module holds view for control objective approval.
```

4.30 my_app.views.control_objective_assessment_view Namespace Reference

Classes

- class ControlObjectiveAssessmentView

4.30.1 Detailed Description

```
This module holds view for control objective assessment.
```

4.31 my_app.views.control_objective_automatization_view Namespace Reference

Classes

- class ControlObjectiveAutomatizationView

4.31.1 Detailed Description

```
This module holds view for control objective automatization.
```

4.32 my_app.views.control_objective_create_view Namespace Reference

Classes

- class ControlObjectiveCreateView

4.32.1 Detailed Description

```
This module holds view for control objective create.
```

4.33 my_app.views.control_objective_delete_view Namespace Reference

Classes

- class ControlObjectiveDeleteView

4.33.1 Detailed Description

```
This module holds view for control objective delete.
```

4.34 my_app.views.control_objective_detail_view Namespace Reference

Classes

- class ControlObjectiveDetailView

4.34.1 Detailed Description

```
This module holds view for control objective detail.
```

4.35 my_app.views.control_objective_edit_view Namespace Reference

Classes

- class ControlObjectiveEditView

4.35.1 Detailed Description

```
This module holds view for control objective edit.
```

4.36 my_app.views.control_objective_generate_view Namespace Reference

Classes

- class GenerateChooseView
- class GenerateForControlQuestionView
- class GenerateForControlView

Functions

- def create_control_objectives (generated_descriptions, form, control, control_question=None)

4.36.1 Detailed Description

This module holds views for automatic generation of control objectives.

4.36.2 Function Documentation

4.36.2.1 create_control_objectives()

```
def my_app.views.control_objective_generate_view.create_control_objectives (
              generated_descriptions,
              form,
              control,
              control_question = None )
```

This method creates control objectives in atomic transaction.
In case of unique constraint violation,
error messages are returned.

4.37 my_app.views.control_question_approval_view Namespace Reference

Classes

- class ControlQuestionApprovalView

4.37.1 Detailed Description

This module holds view for control question approval.

4.38 my_app.views.control_question_assessment_view Namespace Reference

Classes

- class ControlQuestionAssessmentView

4.38.1 Detailed Description

This module holds view for control question assessment.

4.39 my_app.views.control_question_create_view Namespace Reference

Classes

- class ControlQuestionCreateView

4.39.1 Detailed Description

```
This module holds view for control question create.
```

4.40 my_app.views.control_question_delete_view Namespace Reference

Classes

- class ControlQuestionDeleteView

4.40.1 Detailed Description

```
This module holds view for control question delete.
```

4.41 my_app.views.control_question_edit_view Namespace Reference

Classes

- class ControlQuestionEditView

4.41.1 Detailed Description

```
This module holds view for control_question edit.
```

4.42 my_app.views.elements_choose_view Namespace Reference

Classes

- class ElementsChooseView

4.42.1 Detailed Description

```
This module holds view for scheme choose.
```

4.43 my_app.views.error_pages_view Namespace Reference

Functions

- def error_404 (request)
- def error_500 (request)
- def error_404_demo (request)
- def error_500_demo (request)

4.43.1 Detailed Description

This module holds view for custom error pages.

4.43.2 Function Documentation

4.43.2.1 error_404()

```
def my_app.views.error_pages_view.error_404 (
              request )
```

Method for handling error 404

4.43.2.2 error_404_demo()

```
def my_app.views.error_pages_view.error_404_demo (
              request )
```

Method for handling error demo 404

4.43.2.3 error_500()

```
def my_app.views.error_pages_view.error_500 (
              request )
```

Method for handling error 500

4.43.2.4 error_500_demo()

```
def my_app.views.error_pages_view.error_500_demo (
              request )
```

Method for handling error demo 404

4.44 my_app.views.home_page_view Namespace Reference

Classes

- class HomePageView

4.44.1 Detailed Description

```
This module holds view for home page.
```

4.45 my_app.views.metric_approval_view Namespace Reference

Classes

- class MetricApprovalView

4.45.1 Detailed Description

```
This module holds view for metric approval.
```

4.46 my_app.views.metric_change_view Namespace Reference

Classes

- class MetricChangeView

4.46.1 Detailed Description

```
This module holds view for metric change.
```

4.47 my_app.views.my_login_view Namespace Reference

Classes

- class MyLoginView

4.47.1 Detailed Description

```
This module holds view for my login.
```

4.48 my_app.views.my_registration_view Namespace Reference

Classes

- class MyRegistrationView

4.48.1 Detailed Description

```
This module holds view for registration.
```

4.49 my_app.views.password_reset_view Namespace Reference

Classes

- class PasswordResetView

4.49.1 Detailed Description

```
This module holds view for password reset.
```

4.50 my_app.views.public_view Namespace Reference

Functions

- def get_viewable_schemes_rights (user)
- def get_viewable_schemes ()
- def get_assessable_metrics ()
- def get_assessable_security_attributes ()
- def get_security_attribute (request)
- def get_metric (request)
- def get_control_domain (request)
- def get_metrics (request)
- def get_security_attributes (request)
- def get_similar_metrics_by_desc (request)
- def get_similar_metrics_by_name (request)
- def get_similar_security_attributes_by_desc (request)
- def get_similar_security_attributes_by_name (request)
- def get_control_objective (request)
- def describe_security_attributes (request)

4.50.1 Detailed Description

```
This module holds all public methods used by the other views.
```

4.50.2 Function Documentation

4.50.2.1 describe_security_attributes()

```
def my_app.views.public_view.describe_security_attributes (
              request )
```

Method for describing security attributes

4.50.2.2 get_assessable_metrics()

```
def my_app.views.public_view.get_assessable_metrics ( )
```

Method to get metrics, that can be approved, rejected or changed.

4.50.2.3 get_assessable_security_attributes()

```
def my_app.views.public_view.get_assessable_security_attributes ( )
```

Method to get security attributes,
that can be approved, rejected or changed.

4.50.2.4 get_control_domain()

```
def my_app.views.public_view.get_control_domain (
              request )
```

Get control domain for select

4.50.2.5 get_control_objective()

```
def my_app.views.public_view.get_control_objective (
              request )
```

Get control objective for scheme compare detail

4.50.2.6 get_metric()

```
def my_app.views.public_view.get_metric (
              request )
```

Get metric for select

4.50.2.7 get_metrics()

```
def my_app.views.public_view.get_metrics (
              request )
```

Get available metrics for current search

4.50.2.8 get_security_attribute()

```
def my_app.views.public_view.get_security_attribute (
              request )
```

Get security attribute for select.

4.50.2.9 get_security_attributes()

```
def my_app.views.public_view.get_security_attributes (
              request )
```

Get available security attributes for current search

4.50.2.10 get_similar_metrics_by_desc()

```
def my_app.views.public_view.get_similar_metrics_by_desc (
              request )
```

Get similar metrics by description for current metric

4.50.2.11 get_similar_metrics_by_name()

```
def my_app.views.public_view.get_similar_metrics_by_name (
              request )
```

Get similar metrics by name for current metric

4.50.2.12 get_similar_security_attributes_by_desc()

```
def my_app.views.public_view.get_similar_security_attributes_by_desc (
              request )
```

Get similar security attributes by description
for current security_attributes

4.50.2.13 get_similar_security_attributes_by_name()

```
def my_app.views.public_view.get_similar_security_attributes_by_name (
              request )
```

Get similar security attributes by name
for current security attributes

4.50.2.14 get_viewable_schemes()

```
def my_app.views.public_view.get_viewable_schemes ( )
```

Method to get schemes, that can be viewed.

4.50.2.15 get_viewable_schemes_rights()

```
def my_app.views.public_view.get_viewable_schemes_rights (
              user )
```

Method to get ids of the schemes the user can view.

4.51 my_app.views.rdf_export_view Namespace Reference

Classes

- class RdfExportView

4.51.1 Detailed Description

```
This module holds view for rdf export.
```

4.52 my_app.views.scheme_approval_view Namespace Reference

Classes

- class SchemeApprovalView

4.52.1 Detailed Description

```
This module holds view for scheme approval.
```

4.53 my_app.views.scheme_compare_choose_view Namespace Reference

Classes

- class SchemeCompareChooseView

4.53.1 Detailed Description

```
This module holds view for scheme compare choose.
```

4.54 my_app.views.scheme_compare_view Namespace Reference

Classes

- class SchemeCompareView

Functions

- def xstr (s)
- def compare_sentences (sentence_1, sentence_2)
- def compare_type_values (security_attribute_1, security_attribute_2)

4.54.1 Detailed Description

```
This module holds view for scheme compare.
```

4.54.2 Function Documentation

4.54.2.1 compare_sentences()

```
def my_app.views.scheme_compare_view.compare_sentences (
              sentence_1,
              sentence_2 )

Method for comparing sentences
```

4.54.2.2 compare_type_values()

```
def my_app.views.scheme_compare_view.compare_type_values (
              security_attribute_1,
              security_attribute_2 )

Method for comparing values
```

4.54.2.3 xstr()

```
def my_app.views.scheme_compare_view.xstr (
              s )

Method for something
```

4.55 my_app.views.scheme_create_view Namespace Reference

Classes

- class SchemeCreateView

4.55.1 Detailed Description

```
This module holds view for scheme create.
```

4.56 my_app.views.scheme_delete_view Namespace Reference

Classes

- class SchemeDeleteView

4.56.1 Detailed Description

```
This module holds view for scheme delete.
```

4.57 my_app.views.scheme_detail_view Namespace Reference

Classes

- class SchemeDetailView

4.57.1 Detailed Description

```
This module holds view for scheme detail.
```

4.58 my_app.views.scheme_edit_view Namespace Reference

Classes

- class SchemeEditView

4.58.1 Detailed Description

```
This module holds view for scheme edit.
```

4.59 my_app.views.scheme_import_view Namespace Reference

Classes

- class SchemeImportView

4.59.1 Detailed Description

```
This module holds view for scheme import.
```

4.60 my_app.views.scheme_overview_view Namespace Reference

Classes

- class SchemeOverviewView

4.60.1 Detailed Description

```
This module holds view for scheme overview.
```

4.61 my_app.views.scheme_rights_set_view Namespace Reference

Classes

- class SchemeRightsSetView

4.61.1 Detailed Description

```
This module holds view for scheme edit rights set.
```

4.62 my_app.views.security_approval_view Namespace Reference

Classes

- class SecurityApprovalView

4.62.1 Detailed Description

```
This module holds view for security approval view.
```

4.63 my_app.views.security_change_view Namespace Reference

Classes

- class SecurityChangeView

4.63.1 Detailed Description

```
This module holds view for security change.
```

4.64 my_login_form Namespace Reference

Classes

- class MyLoginForm

4.64.1 Detailed Description

```
This module holds form for my login.
```

4.65 my_register_form Namespace Reference

Classes

- class MyRegistrationForm

4.65.1 Detailed Description

```
This module holds form for registration of users.
```

4.66 password_reset_form Namespace Reference

Classes

- class PasswordResetForm

4.66.1 Detailed Description

```
This module holds form for password reset.
```

4.67 scheme_compare_choose_form Namespace Reference

Classes

- class SchemeCompareChooseForm

4.67.1 Detailed Description

```
This module holds form for scheme compare choose.
```

4.68 scheme_create_form Namespace Reference

Classes

- class SchemeCreateForm

4.68.1 Detailed Description

This module holds form for scheme create.

4.69 scheme_import_form Namespace Reference

Classes

- class SchemeImportForm

4.69.1 Detailed Description

This module holds form for scheme import.

4.70 scheme_rights_set_form Namespace Reference

Classes

- class SchemeRightsSetForm

4.70.1 Detailed Description

This module holds form for scheme rights set.

4.71 security_change_form Namespace Reference

Classes

- class SecurityChangeForm

4.71.1 Detailed Description

This module holds form for security change.

Chapter 5

Class Documentation

5.1 my_app.models.CertificationScheme Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)
- def can_assess (self)
- def get_viewable_controls (self)
- def get_assessable_controls (self)
- def get_unassessable_controls (self)
- def get_generateable_controls (self)
- def get_generateable_control_questions (self)
- def can_view_rights (self, user)
- def can_edit_rights (self, user)
- def scheme_status (self, user)
- def can_review_rights (self, user)
- def can_setrights_rights (self, user)
- def can_publish_rights (self, user)

Static Public Attributes

- **scheme_name**
- **max_length**
- **identifier**
- **version**
- **publisher**
- **NEW**
- **APPROVED**
- **REJECTED**

- **REWORK**
- **PUBLISHED**
- **STATUS_CHOICES**
- **CAN_VIEW_CHOICES**
- **CAN_EDIT_CHOICES**
- **CAN_DELETE_CHOICES**
- **CAN_REVIEW_CHOICES**
- **CAN_SET_RIGHTS_CHOICES**
- **CAN_PUBLISH_CHOICES**
- **status**
- **choices**
- **default**

5.1.1 Detailed Description

```
Model for certification scheme
```

5.1.2 Member Function Documentation

5.1.2.1 can_assess()

```
def my_app.models.CertificationScheme.can_assess (
              self )
```

```
Method to check if this scheme can be assessed.
```

5.1.2.2 can_edit_rights()

```
def my_app.models.CertificationScheme.can_edit_rights (
              self,
              user )
```

```
Method to check if user can
edit this certification scheme.
```

5.1.2.3 can_publish_rights()

```
def my_app.models.CertificationScheme.can_publish_rights (
              self,
              user )
```

Method to check if user can
publish this certification scheme.

5.1.2.4 can_review_rights()

```
def my_app.models.CertificationScheme.can_review_rights (
              self,
              user )
```

Method to check if user can
review this certification scheme.

5.1.2.5 can_setrights_rights()

```
def my_app.models.CertificationScheme.can_setrights_rights (
              self,
              user )
```

Method to check if user can
set rights to this certification scheme.

5.1.2.6 can_view_rights()

```
def my_app.models.CertificationScheme.can_view_rights (
              self,
              user )
```

Method to check if user can
view this certification scheme.

5.1.2.7 get_assessable_controls()

```
def my_app.models.CertificationScheme.get_assessable_controls (
              self )
```

Method to get assessable controls.

5.1.2.8 get_generateable_control_questions()

```
def my_app.models.CertificationScheme.get_generateable_control_questions (
              self )
```

Method to get control questions,
that control objectives can be generated for.

5.1.2.9 get_generateable_controls()

```
def my_app.models.CertificationScheme.get_generateable_controls (
              self )
```

Method to get controls,
that control objectives can be generated for.

5.1.2.10 get_unassessable_controls()

```
def my_app.models.CertificationScheme.get_unassessable_controls (
              self )
```

Method to get unassessable controls.

5.1.2.11 get_viewable_controls()

```
def my_app.models.CertificationScheme.get_viewable_controls (
              self )
```

Method to get viewable controls.

5.1.2.12 scheme_status()

```
def my_app.models.CertificationScheme.scheme_status (
              self,
              user )
```

Method to return role of user for scheme.

5.2 my_app.models.CloudService Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **provider**
- **CloudServiceProvider**
- **on_delete**
- **service_name**
- **max_length**
- **unique**
- **description**
- **IAAS**
- **SAAS**
- **PAAS**
- **MBAAS**
- **SERVERLESS**
- **SERVICE_CHOICES**
- **service_model**
- **choices**
- **default**
- **PRIVATE**
- **PUBLIC**
- **HYBRID**
- **COMMUNITY**
- **DISTRIBUTED**
- **MULTI**
- **BIGDATA**
- **HPC**
- **DEPLOYMENT_CHOICES**
- **deployment_model**

5.2.1 Detailed Description

```
Model for cloud service
```

5.3 cloud_service_create_form.CloudServiceCreateForm Class Reference

Inherits Form.

Static Public Attributes

- **service_name**
- **description**
- **service_model**
- **deployment_model**

5.3.1 Detailed Description

```
Form for creating cloud service.
```

5.3.2 Member Data Documentation

5.3.2.1 deployment_model

```
cloud_service_create_form.CloudServiceCreateForm.deployment_model  [static]
```

Initial value:

```
=  forms.ChoiceField(choices=CloudService.DEPLOYMENT_CHOICES, required=True,
                                        label=_("Deployment model"))
```

5.3.2.2 description

```
cloud_service_create_form.CloudServiceCreateForm.description  [static]
```

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Description"),
        max_length=CloudService._meta.get_field("description").max_length
    )
```

5.3.2.3 service_model

```
cloud_service_create_form.CloudServiceCreateForm.service_model  [static]
```

Initial value:

```
=  forms.ChoiceField(choices=CloudService.SERVICE_CHOICES, required=True,
                                    label=_("Service model"))
```

5.3.2.4 service_name

```
cloud_service_create_form.CloudServiceCreateForm.service_name  [static]
```

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Cloud service name"),
        max_length=CloudService._meta.get_field("service_name").max_length
    )
```

5.4 my_app.views.cloud_service_create_view.CloudServiceCreateView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = \

5.4.1 Detailed Description

```
Class-based view for cloud service creation.
```

5.4.2 Member Function Documentation

5.4.2.1 get()

```
def my_app.views.cloud_service_create_view.CloudServiceCreateView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.4.2.2 post()

```
def my_app.views.cloud_service_create_view.CloudServiceCreateView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect
after cloud service creation.

5.5 my_app.models.CloudServiceProvider Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **user**
- **User**
- **on_delete**
- **provider_name**
- **max_length**
- **unique**

5.5.1 Detailed Description

```
Model for cloud service provider
```

5.6 cloud_service_provider_register_form.CloudServiceProviderRegisterForm Class Reference

Inherits Form.

Static Public Attributes

- **provider_name**
- **next** = forms.CharField(widget=forms.HiddenInput(), required=False)

5.6.1 Detailed Description

```
Form for cloud service provider registration.
```

5.6.2 Member Data Documentation

5.6.2.1 provider_name

```
cloud_service_provider_register_form.CloudServiceProviderRegisterForm.provider_name  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": ""}),
        label=_("Provider name"),
        max_length=CloudServiceProvider._meta.get_field(
            "provider_name").max_length)
```

5.7 my_app.views.cloud_service_provider_register_view.CloudServiceProviderRegisterView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = \

5.7.1 Detailed Description

```
Class-based view for cloud service provider registration.
```

5.7.2 Member Function Documentation

5.7.2.1 get()

```
def my_app.views.cloud_service_provider_register_view.CloudServiceProviderRegisterView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.7.2.2 post()

```
def my_app.views.cloud_service_provider_register_view.CloudServiceProviderRegisterView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after
cloud service provider registration.
```

5.8 my_app.views.cloud_service_overview_view.CloudServicesOverviewView Class Reference

Inherits TemplateView.

Public Member Functions

- def get (self, request, args, kwargs)

5.8.1 Detailed Description

Class-based view for cloud services overview.

5.8.2 Member Function Documentation

5.8.2.1 get()

```
def my_app.views.cloud_service_overview_view.CloudServicesOverviewView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.9 comment_form.CommentForm Class Reference

Inherits Form.

Static Public Attributes

- **comment_text**

5.9.1 Detailed Description

Form for comments

5.9.2 Member Data Documentation

5.9.2.1 comment_text

comment_form.CommentForm.comment_text [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={
            "placeholder": _("Place for comment")}),
        label=_("Comment"),
        max_length=ControlComment._meta.get_field("comment_text").max_length)
```

5.10 my_app.models.Control Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)
- def get_viewable_control_questions (self)
- def get_automatization_viewable_control_objectives (self)
- def get_viewable_control_objectives (self)
- def get_automatization_control_objectives (self)
- def get_assessable_control_objectives (self)
- def get_unassessable_control_objectives (self)
- def get_assessable_control_questions (self)
- def get_unassessable_control_questions (self)

Static Public Attributes

- **scheme**
- **CertificationScheme**
- **on_delete**
- **identifier**
- **max_length**
- **description**
- **NEW**
- **APPROVED**
- **REJECTED**
- **REWORK**
- **STATUS_CHOICES**
- **CAN_VIEW_CHOICES**
- **CAN_EDIT_CHOICES**
- **CAN_DELETE_CHOICES**
- **CAN_REVIEW_CHOICES**
- **status**
- **choices**
- **default**

5.10.1 Detailed Description

```
Model for control
```

5.10.2 Member Function Documentation

5.10.2.1 get_assessable_control_objectives()

```
def my_app.models.Control.get_assessable_control_objectives (
              self )
```

Method to get assessable control objectives.

5.10.2.2 get_assessable_control_questions()

```
def my_app.models.Control.get_assessable_control_questions (
              self )
```

Method to get assessable control questions.

5.10.2.3 get_automatization_control_objectives()

```
def my_app.models.Control.get_automatization_control_objectives (
              self )
```

Method to get automatically generateable control objectives.

5.10.2.4 get_automatization_viewable_control_objectives()

```
def my_app.models.Control.get_automatization_viewable_control_objectives (
              self )
```

Method to get automatically generateable and viewable control objectives.

5.10.2.5 get_unassessable_control_objectives()

```
def my_app.models.Control.get_unassessable_control_objectives (
              self )
```

Method to get unassessable control objectives.

5.10.2.6 get_unassessable_control_questions()

```
def my_app.models.Control.get_unassessable_control_questions (
              self )
```

Method to get unassessable control objectives.

5.10.2.7 get_viewable_control_objectives()

```
def my_app.models.Control.get_viewable_control_objectives (
              self )
```

Method to get viewable control objectives.

5.10.2.8 get_viewable_control_questions()

```
def my_app.models.Control.get_viewable_control_questions (
              self )
```

Method to get viewable control questions.

5.11 my_app.views.control_approval_view.ControlApprovalView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = comment_form.CommentForm

5.11.1 Detailed Description

Class-based view for control approval.

5.11.2 Member Function Documentation

5.11.2.1 get()

```
def my_app.views.control_approval_view.ControlApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.11.2.2 post()

```
def my_app.views.control_approval_view.ControlApprovalView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method. Returns HttpResponse for input request.

5.12 my_app.views.control_assessment_view.ControlAssessmentView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.12.1 Detailed Description

Class-based view for control assessment.

5.12.2 Member Function Documentation

5.12.2.1 get()

```
def my_app.views.control_assessment_view.ControlAssessmentView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.13 my_app.models.ControlComment Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **control**
- **author**
- **User**
- **default**
- **comment_text**
- **max_length**
- **date**

5.13.1 Detailed Description

```
Model for control comments
```

5.14 control_create_form.ControlCreateForm Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**

5.14.1 Detailed Description

Form for control creation.

5.14.2 Member Data Documentation

5.14.2.1 description

control_create_form.ControlCreateForm.description [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Description"),
        max_length=Control._meta.get_field("description").max_length)
```

5.14.2.2 identifier

control_create_form.ControlCreateForm.identifier [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Identifier"),
        max_length=Control._meta.get_field("identifier").max_length)
```

5.15 my_app.views.control_create_view.ControlCreateView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_create_form.ControlCreateForm

5.15.1 Detailed Description

Class-based view for control creation.

5.15.2 Member Function Documentation

5.15.2.1 get()

```
def my_app.views.control_create_view.ControlCreateView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.15.2.2 post()

```
def my_app.views.control_create_view.ControlCreateView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control creation.

5.16 my_app.views.control_delete_view.ControlDeleteView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.16.1 Detailed Description

Class-based view for control delete.

5.16.2 Member Function Documentation

5.16.2.1 get()

```
def my_app.views.control_delete_view.ControlDeleteView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.17 my_app.views.control_detail_view.ControlDetailView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.17.1 Detailed Description

Class-based view for control detail.

5.17.2 Member Function Documentation

5.17.2.1 get()

```
def my_app.views.control_detail_view.ControlDetailView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.18 my_app.models.ControlDomain Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **control_domain_name**
- **max_length**
- **description**
- **unique**

5.18.1 Detailed Description

```
Model for control domain
```

5.19 control_domain_select_form.ControlDomainSelectForm Class Reference

Inherits Form.

Static Public Attributes

- **control_domain**
- **control_domain_details**

5.19.1 Detailed Description

```
Form for control domain selection
```

5.19.2 Member Data Documentation

5.19.2.1 control_domain

```
control_domain_select_form.ControlDomainSelectForm.control_domain  [static]
```

Initial value:

```
=  forms.ModelChoiceField(
        required=False,
        widget=forms.Select(),
        queryset=ControlDomain.objects.all(),
        label=_("Control domain"))
```

5.19.2.2 control_domain_details

```
control_domain_select_form.ControlDomainSelectForm.control_domain_details  [static]
```

Initial value:

```
=  forms.BooleanField(
        required=False,
        label=_("Control domain details"),
        widget=forms.CheckboxInput(),
        initial=False)
```

5.20 my_app.views.control_edit_view.ControlEditView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_create_form.ControlCreateForm

5.20.1 Detailed Description

```
Class-based view for control edit.
```

5.20.2 Member Function Documentation

5.20.2.1 get()

```
def my_app.views.control_edit_view.ControlEditView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.20.2.2 post()

```
def my_app.views.control_edit_view.ControlEditView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control edit.

5.21 my_app.models.ControlObjective Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)
- def get_value (self)

Static Public Attributes

- **control**
- **Control**
- **on_delete**
- **control_question**
- **ControlQuestion**
- **null**
- **True**
- **security_attribute**
- **SecurityAttribute**
- **identifier**
- **max_length**
- **unique**
- **description**
- **value_min**
- **blank**
- **value_max**
- **evaluation_interval**
- **NEW**
- **APPROVED**
- **REJECTED**
- **REWORK**
- **GENERATED**
- **STATUS_CHOICES**

- **CAN_VIEW_CHOICES**
- **CAN_EDIT_CHOICES**
- **CAN_AUTOMATIZATION_CHOICES**
- **CAN_DELETE_CHOICES**
- **CAN_REVIEW_CHOICES**
- **status**
- **choices**
- **default**
- **NONE**
- **GV**
- **MIN_GV**
- **MAX_GV**
- **BETWEEN_GV**
- **TYPE_CHOICES**
- **type**

5.21.1 Detailed Description

```
Model for control objective
```

5.21.2 Member Function Documentation

5.21.2.1 get_value()

```
def my_app.models.ControlObjective.get_value (
              self )
```

```
This method returns value according to type.
```

5.22 my_app.views.control_objective_approval_view.ControlObjectiveApprovalView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = comment_form.CommentForm

5.22.1 Detailed Description

```
Class-based view for control objective approval
```

5.22.2 Member Function Documentation

5.22.2.1 get()

```
def my_app.views.control_objective_approval_view.ControlObjectiveApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.22.2.2 post()

```
def my_app.views.control_objective_approval_view.ControlObjectiveApprovalView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method. Returns HttpResponse for input request.
```

5.23 my_app.views.control_objective_assessment_view.ControlObjectiveAssessment↩ View Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.23.1 Detailed Description

```
Class-based view for control objective assessment
```

5.23.2 Member Function Documentation

5.23.2.1 get()

```
def my_app.views.control_objective_assessment_view.ControlObjectiveAssessmentView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.24 control_objective_automatization_form.ControlObjectiveAutomatizationForm Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**

5.24.1 Detailed Description

Form for control objective automatization.

5.24.2 Member Data Documentation

5.24.2.1 description

control_objective_automatization_form.ControlObjectiveAutomatizationForm.description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of control objective")}),
        label=_("Description"),
        max_length=ControlObjective._meta.get_field("description").max_length)
```

5.24.2.2 identifier

```
control_objective_automatization_form.ControlObjectiveAutomatizationForm.identifier  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Identifier of control objective"),
                   "size": 40}),
        label=_("Identifier"),
        max_length=ControlObjective._meta.get_field("identifier").max_length)
```

5.25 my_app.views.control_objective_automatization_view.ControlObjectiveAutomatization↩ View Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Member Functions

- def update_control_objective (user, form, control_objective)

Static Public Attributes

- **form_class** = control_objective_automatization_form.\
- **sa_search_class** = control_objective_automatization_form.\
- **sa_recommend_class** = control_objective_automatization_form.\
- **sa_generate_class** = control_objective_automatization_form.\
- **met_sa_search_class** = control_objective_automatization_form.\
- **met_sa_recommend_class** = control_objective_automatization_form.\
- **met_search_class** = control_objective_automatization_form.\
- **met_generate_class** = control_objective_automatization_form.\
- **met_recommend_class** = control_objective_automatization_form.\
- **form_rest_class** = control_objective_automatization_form.\
- **form_save_class** = control_objective_automatization_form.\

5.25.1 Detailed Description

```
Class-based view for control objective automatization.
```

5.25.2 Member Function Documentation

5.25.2.1 get()

```
def my_app.views.control_objective_automatization_view.ControlObjectiveAutomatizationView.get
(
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.25.2.2 post()

```
def my_app.views.control_objective_automatization_view.ControlObjectiveAutomatizationView.post
(
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control objective creation.

5.25.2.3 update_control_objective()

```
def my_app.views.control_objective_automatization_view.ControlObjectiveAutomatizationView.↩
update_control_objective (
              user,
              form,
              control_objective )  [static]
```

This method creates security attribute,
metric and control objective in atomic transaction.
In case of unique constraint violation,
error messages are returned.

5.26 my_app.models.ControlObjectiveComment Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **control_objective**
- **author**
- **User**
- **default**
- **comment_text**
- **max_length**
- **date**

5.26.1 Detailed Description

```
Model for control objective comments
```

5.27 control_objective_create_form.ControlObjectiveCreateForm Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**
- **new_security_attribute**
- **find_security_attribute**
- **security_attribute**
- **security_attribute_name**
- **security_attribute_description**
- **new_metric**
- **find_metric**
- **metric**
- **metric_name**
- **metric_description**
- **metric_expression**
- **metric_measuring_interval**
- **control_domain**
- **type**
- **type_value**
- **type_value_2**
- **evaluation_interval**

5.27.1 Detailed Description

```
Form for control objective creation.
```

5.27.2 Member Data Documentation

5.27.2.1 control_domain

control_objective_create_form.ControlObjectiveCreateForm.control_domain [static]

Initial value:

```
=  forms.ModelChoiceField(
            widget=forms.Select(),
            queryset=ControlDomain.objects.all(),
            empty_label=None,
            label=_("Security attribute control domain"))
```

5.27.2.2 description

control_objective_create_form.ControlObjectiveCreateForm.description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of control objective")}),
        label=_("Description"),
        max_length=ControlObjective._meta.get_field("description").max_length)
```

5.27.2.3 evaluation_interval

control_objective_create_form.ControlObjectiveCreateForm.evaluation_interval [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder":
                       _("Evaluation interval of control objective"),
                   "size": 45}),
        label=_("Evaluation interval"),
        max_length=ControlObjective._meta.get_field(
            "evaluation_interval").max_length)
```

5.27.2.4 find_metric

control_objective_create_form.ControlObjectiveCreateForm.find_metric [static]

Initial value:

```
=  forms.CharField(
        required=False,
        widget=forms.TextInput(
            attrs={"placeholder": _("Search metric..."), "size": 40}),
        label=_("Metric Search"))
```

5.27.2.5 find_security_attribute

control_objective_create_form.ControlObjectiveCreateForm.find_security_attribute [static]

Initial value:

```
=  forms.CharField(
        required=False,
        widget=forms.TextInput(
            attrs={"placeholder": _("Search security attribute..."),
                   "size": 40}),
        label=_("Security Attribute Search"))
```

5.27.2.6 identifier

control_objective_create_form.ControlObjectiveCreateForm.identifier [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Identifier of control objective"),
                   "size": 40}),
        label=_("Identifier"),
        max_length=ControlObjective._meta.get_field("identifier").max_length)
```

5.27.2.7 metric

control_objective_create_form.ControlObjectiveCreateForm.metric [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        queryset=Metric.objects.all(),
        empty_label=None,
        label=_("Security attribute metric"))
```

5.27.2.8 metric_description

control_objective_create_form.ControlObjectiveCreateForm.metric_description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        max_length=Metric._meta.get_field("description").max_length)
```

5.27.2.9 metric_expression

control_objective_create_form.ControlObjectiveCreateForm.metric_expression [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        max_length=Metric._meta.get_field("expression").max_length)
```

5.27.2.10 metric_measuring_interval

control_objective_create_form.ControlObjectiveCreateForm.metric_measuring_interval [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"))
```

5.27.2.11 metric_name

control_objective_create_form.ControlObjectiveCreateForm.metric_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        max_length=Metric._meta.get_field("metric_name").max_length)
```

5.27.2.12 new_metric

control_objective_create_form.ControlObjectiveCreateForm.new_metric [static]

Initial value:

```
=  forms.BooleanField(
        required=False,
        widget=forms.CheckboxInput(),
        label=_("New metric"))
```

5.27.2.13 new_security_attribute

control_objective_create_form.ControlObjectiveCreateForm.new_security_attribute [static]

Initial value:

```
=  forms.BooleanField(
        required=False,
        widget=forms.CheckboxInput(),
        label=_("New security attribute"))
```

5.27.2.14 security_attribute

control_objective_create_form.ControlObjectiveCreateForm.security_attribute [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        queryset=SecurityAttribute.objects.all(),
        empty_label=None,
        label=_("Security attribute"))
```

5.27.2.15 security_attribute_description

control_objective_create_form.ControlObjectiveCreateForm.security_attribute_description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of security attribute")}),
        label=_("Security attribute description"),
        max_length=SecurityAttribute._meta.get_field("description").max_length)
```

5.27.2.16 security_attribute_name

control_objective_create_form.ControlObjectiveCreateForm.security_attribute_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Name of security attribute")}),
        label=_("Security attribute name"),
        max_length=SecurityAttribute._meta.get_field(
            "security_attribute_name").max_length)
```

5.27.2.17 type

control_objective_create_form.ControlObjectiveCreateForm.type [static]

Initial value:

```
=  forms.ChoiceField(
        choices=ControlObjective.TYPE_CHOICES,
        label=_("Type"),
        initial="",
        widget=forms.Select())
```

5.27.2.18 type_value

control_objective_create_form.ControlObjectiveCreateForm.type_value [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Value for chosen type"), "size": 30}),
        label=_("Type value"),
        max_length=max(ControlObjective._meta.get_field(
            "value_min").max_length,
                       ControlObjective._meta.get_field(
                           "value_max").max_length))
```

5.27.2.19 type_value_2

control_objective_create_form.ControlObjectiveCreateForm.type_value_2 [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Second value for chosen type"),
                   "size": 30}),
        label=_("Second type value"),
        required=False,
        max_length=ControlObjective._meta.get_field("value_max").max_length)
```

5.28 my_app.views.control_objective_create_view.ControlObjectiveCreateView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Member Functions

- def get_control_objective_type (control_objective_type, type_value, type_value_2)
- def validate_unique (control_objective)
- def save_atomic_control_objective (control_objective, security_attribute, metric)
- def create_control_objective (user, form, control)

Static Public Attributes

- **form_class** = control_objective_create_form.ControlObjectiveCreateForm

5.28.1 Detailed Description

```
Class-based view for control objective creation.
```

5.28.2 Member Function Documentation

5.28.2.1 create_control_objective()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.create_control_↩
objective (
              user,
              form,
              control )  [static]
```

```
This method creates security attribute,
metric and control objective in atomic transaction.
In case of unique constraint violation,
error messages are returned.
```

5.28.2.2 get()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.28.2.3 get_control_objective_type()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.get_control_objective↩
_type (
              control_objective_type,
              type_value,
              type_value_2 )  [static]
```

This method returns min/max value according to type.

5.28.2.4 post()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control objective creation.

5.28.2.5 save_atomic_control_objective()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.save_atomic_control↩
_objective (
              control_objective,
              security_attribute,
              metric )  [static]
```

This method creates metric, security attribute
and control objective in atomic transaction.

5.28.2.6 validate_unique()

```
def my_app.views.control_objective_create_view.ControlObjectiveCreateView.validate_unique (
              control_objective )  [static]
```

```
This method validates unique constraint
for (identifier) and (scheme, security attribute).
```

5.29 my_app.views.control_objective_delete_view.ControlObjectiveDeleteView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.29.1 Detailed Description

```
Class-based view for control objective delete.
```

5.29.2 Member Function Documentation

5.29.2.1 get()

```
def my_app.views.control_objective_delete_view.ControlObjectiveDeleteView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.30 my_app.views.control_objective_detail_view.ControlObjectiveDetailView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.30.1 Detailed Description

```
Class-based view for control detail.
```

5.30.2 Member Function Documentation

5.30.2.1 get()

```
def my_app.views.control_objective_detail_view.ControlObjectiveDetailView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.31 control_objective_edit_form.ControlObjectiveEditForm Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**
- **type**
- **type_value**
- **type_value_2**
- **evaluation_interval**

5.31.1 Detailed Description

```
Form for control objective creation.
```

5.31.2 Member Data Documentation

5.31.2.1 description

control_objective_edit_form.ControlObjectiveEditForm.description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of control objective")}),
        label=_("Description"),
        max_length=ControlObjective._meta.get_field("description").max_length)
```

5.31.2.2 evaluation_interval

control_objective_edit_form.ControlObjectiveEditForm.evaluation_interval [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder":
                       _("Evaluation interval of control objective"),
                   "size": 45}),
        label=_("Evaluation interval"),
        max_length=ControlObjective._meta.get_field(
            "evaluation_interval").max_length)
```

5.31.2.3 identifier

control_objective_edit_form.ControlObjectiveEditForm.identifier [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Identifier of control objective"),
                   "size": 40}),
        label=_("Identifier"),
        max_length=ControlObjective._meta.get_field("identifier").max_length)
```

5.31.2.4 type

control_objective_edit_form.ControlObjectiveEditForm.type [static]

Initial value:

```
=  forms.ChoiceField(
        choices=ControlObjective.TYPE_CHOICES,
        label=_("Type"),
        initial="",
        widget=forms.Select())
```

5.31.2.5 type_value

control_objective_edit_form.ControlObjectiveEditForm.type_value [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Value for chosen type"), "size": 30}),
        label=_("Type value"),
        max_length=max(ControlObjective._meta.get_field(
            "value_min").max_length,
                       ControlObjective._meta.get_field(
                           "value_max").max_length))
```

5.31.2.6 type_value_2

control_objective_edit_form.ControlObjectiveEditForm.type_value_2 [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Second value for chosen type"),
                   "size": 30}),
        label=_("Second type value"),
        required=False,
        max_length=ControlObjective._meta.get_field("value_max").max_length)
```

5.32 my_app.views.control_objective_edit_view.ControlObjectiveEditView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_objective_edit_form.ControlObjectiveEditForm

5.32.1 Detailed Description

Class-based view for control objective edit.

5.32.2 Member Function Documentation

5.32.2.1 get()

```
def my_app.views.control_objective_edit_view.ControlObjectiveEditView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.32.2.2 post()

```
def my_app.views.control_objective_edit_view.ControlObjectiveEditView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after control objective edit.
```

5.33 control_objective_generate_form.ControlObjectiveGenerateForm Class Reference

Inherits Form.

Public Member Functions

- def **__init__** (self, args, kwargs)

5.33.1 Detailed Description

```
Form for inspecting and creating control objectives.
```

5.34 control_objective_automatization_form.ControlObjectiveRemainsForm Class Reference

Inherits Form.

Static Public Attributes

- **control_domain**
- **type**
- **type_value**
- **type_value_2**
- **evaluation_interval**

5.34.1 Detailed Description

```
Form for remaining control objective
```

5.34.2 Member Data Documentation

5.34.2.1 control_domain

```
control_objective_automatization_form.ControlObjectiveRemainsForm.control_domain  [static]
```

Initial value:

```
=  forms.ModelChoiceField(
            widget=forms.Select(),
            queryset=ControlDomain.objects.all(),
            empty_label=None,
            label=_("Security attribute control domain"))
```

5.34.2.2 evaluation_interval

```
control_objective_automatization_form.ControlObjectiveRemainsForm.evaluation_interval  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder":
                       _("Evaluation interval of control objective"),
                   "size": 45}),
        label=_("Evaluation interval"),
        initial="365",
        max_length=ControlObjective._meta.get_field(
            "evaluation_interval").max_length)
```

5.34.2.3 type

control_objective_automatization_form.ControlObjectiveRemainsForm.type [static]

Initial value:

```
=  forms.ChoiceField(
        choices=ControlObjective.TYPE_CHOICES,
        label=_("Type"),
        initial="",
        widget=forms.Select())
```

5.34.2.4 type_value

control_objective_automatization_form.ControlObjectiveRemainsForm.type_value [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Value for chosen type"), "size": 30}),
        label=_("Type value"),
        max_length=max(ControlObjective._meta.get_field(
            "value_min").max_length,
                       ControlObjective._meta.get_field(
                           "value_max").max_length))
```

5.34.2.5 type_value_2

control_objective_automatization_form.ControlObjectiveRemainsForm.type_value_2 [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Second value for chosen type"),
                   "size": 30}),
        label=_("Second type value"),
        required=False,
        max_length=ControlObjective._meta.get_field("value_max").max_length)
```

5.35 my_app.models.ControlQuestion Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **control**
- **Control**
- **on_delete**
- **identifier**
- **max_length**
- **description**
- **NEW**
- **APPROVED**
- **REJECTED**
- **REWORK**
- **STATUS_CHOICES**
- **CAN_VIEW_CHOICES**
- **CAN_EDIT_CHOICES**
- **CAN_DELETE_CHOICES**
- **CAN_REVIEW_CHOICES**
- **status**
- **choices**
- **default**

5.35.1 Detailed Description

```
Model for control question
```

5.36 my_app.views.control_question_approval_view.ControlQuestionApprovalView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = comment_form.CommentForm

5.36.1 Detailed Description

```
Class-based view for control question approval
```

5.36.2 Member Function Documentation

5.36.2.1 get()

```
def my_app.views.control_question_approval_view.ControlQuestionApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.36.2.2 post()

```
def my_app.views.control_question_approval_view.ControlQuestionApprovalView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method. Returns HttpResponse for input request.

5.37 my_app.views.control_question_assessment_view.ControlQuestionAssessmentView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.37.1 Detailed Description

Class-based view for control assessment.

5.37.2 Member Function Documentation

5.37.2.1 get()

```
def my_app.views.control_question_assessment_view.ControlQuestionAssessmentView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.38 my_app.models.ControlQuestionComment Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **control_question**
- **author**
- **User**
- **default**
- **comment_text**
- **max_length**
- **date**

5.38.1 Detailed Description

Model for control question comments

5.39 control_question_create_form.ControlQuestionCreateForm Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**

5.39.1 Detailed Description

Form for control question creation.

5.39.2 Member Data Documentation

5.39.2.1 description

control_question_create_form.ControlQuestionCreateForm.description [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Description"),
        max_length=ControlQuestion._meta.get_field("description").max_length)
```

5.39.2.2 identifier

control_question_create_form.ControlQuestionCreateForm.identifier [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}), label=_("Identifier"),
        max_length=ControlQuestion._meta.get_field("identifier").max_length)
```

5.40 my_app.views.control_question_create_view.ControlQuestionCreateView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_question_create_form.ControlQuestionCreateForm

5.40.1 Detailed Description

```
Class-based view for control question creation.
```

5.40.2 Member Function Documentation

5.40.2.1 get()

```
def my_app.views.control_question_create_view.ControlQuestionCreateView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.40.2.2 post()

```
def my_app.views.control_question_create_view.ControlQuestionCreateView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after control creation.
```

5.41 my_app.views.control_question_delete_view.ControlQuestionDeleteView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.41.1 Detailed Description

```
Class-based view for control question delete.
```

5.41.2 Member Function Documentation

5.41.2.1 get()

```
def my_app.views.control_question_delete_view.ControlQuestionDeleteView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.42 my_app.views.control_question_edit_view.ControlQuestionEditView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_question_create_form.ControlQuestionCreateForm

5.42.1 Detailed Description

Class-based view for control_question edit.

5.42.2 Member Function Documentation

5.42.2.1 get()

```
def my_app.views.control_question_edit_view.ControlQuestionEditView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.42.2.2 post()

```
def my_app.views.control_question_edit_view.ControlQuestionEditView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after control_question edit.
```

5.43 my_app.views.elements_choose_view.ElementsChooseView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.43.1 Detailed Description

```
Class for choosing a scheme for changing
```

5.43.2 Member Function Documentation

5.43.2.1 get()

```
def my_app.views.elements_choose_view.ElementsChooseView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.44 control_objective_automatization_form.FormSave Class Reference

Inherits Form.

Static Public Attributes

- **identifier**
- **description**
- **security_attribute_name**
- **security_attribute_description**
- **metric_name**
- **metric_description**
- **metric_expression**
- **metric_measuring_interval**
- **control_domain**
- **type**
- **type_value**
- **type_value_2**
- **evaluation_interval**

5.44.1 Detailed Description

```
Form for saving
```

5.44.2 Member Data Documentation

5.44.2.1 control_domain

```
control_objective_automatization_form.FormSave.control_domain  [static]
```

Initial value:

```
=  forms.ModelChoiceField(
            widget=forms.Select(),
            queryset=ControlDomain.objects.all(),
            empty_label=None,
            label=_("Security attribute control domain"))
```

5.44.2.2 description

```
control_objective_automatization_form.FormSave.description  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of control objective")}),
        label=_("Description"),
        max_length=ControlObjective._meta.get_field("description").max_length)
```

5.44.2.3 evaluation_interval

control_objective_automatization_form.FormSave.evaluation_interval [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder":
                       _("Evaluation interval of control objective"),
                   "size": 45}),
        label=_("Evaluation interval"),
        max_length=ControlObjective._meta.get_field(
            "evaluation_interval").max_length)
```

5.44.2.4 identifier

control_objective_automatization_form.FormSave.identifier [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Identifier of control objective"),
                   "size": 40}),
        label=_("Identifier"),
        max_length=ControlObjective._meta.get_field("identifier").max_length)
```

5.44.2.5 metric_description

control_objective_automatization_form.FormSave.metric_description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        max_length=Metric._meta.get_field("description").max_length)
```

5.44.2.6 metric_expression

control_objective_automatization_form.FormSave.metric_expression [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        max_length=Metric._meta.get_field("expression").max_length)
```

5.44.2.7 metric_measuring_interval

control_objective_automatization_form.FormSave.metric_measuring_interval [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"))
```

5.44.2.8 metric_name

control_objective_automatization_form.FormSave.metric_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        max_length=Metric._meta.get_field("metric_name").max_length)
```

5.44.2.9 security_attribute_description

control_objective_automatization_form.FormSave.security_attribute_description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of security attribute")}),
        label=_("Security attribute description"),
        max_length=SecurityAttribute._meta.get_field("description").max_length)
```

5.44.2.10 security_attribute_name

control_objective_automatization_form.FormSave.security_attribute_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Name of security attribute")}),
        label=_("Security attribute name"),
        max_length=SecurityAttribute._meta.get_field(
            "security_attribute_name").max_length)
```

5.44.2.11 type

control_objective_automatization_form.FormSave.type [static]

Initial value:

```
=  forms.ChoiceField(
        choices=ControlObjective.TYPE_CHOICES,
        label=_("Type"),
        initial="",
        widget=forms.Select())
```

5.44.2.12 type_value

control_objective_automatization_form.FormSave.type_value [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Value for chosen type"), "size": 30}),
        label=_("Type value"),
        max_length=max(ControlObjective._meta.get_field(
            "value_min").max_length,
                       ControlObjective._meta.get_field(
                           "value_max").max_length))
```

5.44.2.13 type_value_2

control_objective_automatization_form.FormSave.type_value_2 [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Second value for chosen type"),
                   "size": 30}),
        label=_("Second type value"),
        required=False,
        max_length=ControlObjective._meta.get_field("value_max").max_length)
```

5.45 my_app.views.control_objective_generate_view.GenerateChooseView Class Reference

Inherits TemplateView.

Public Member Functions

- def get (self, request, args, kwargs)

5.45.1 Detailed Description

```
View for choosing of control or
control question to generate control
objectives for.
```

5.45.2 Member Function Documentation

5.45.2.1 get()

```
def my_app.views.control_objective_generate_view.GenerateChooseView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.46 my_app.views.control_objective_generate_view.GenerateForControlQuestionView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_objective_generate_form.ControlObjectiveGenerateForm

5.46.1 Detailed Description

```
View for inspecting automatically generated
control objective descriptions for control question.
```

5.46.2 Member Function Documentation

5.46.2.1 get()

```
def my_app.views.control_objective_generate_view.GenerateForControlQuestionView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.46.2.2 post()

```
def my_app.views.control_objective_generate_view.GenerateForControlQuestionView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control objectives creation.

5.47 my_app.views.control_objective_generate_view.GenerateForControlView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = control_objective_generate_form.ControlObjectiveGenerateForm

5.47.1 Detailed Description

View for inspecting automatically
generated control objective descriptions for control.

5.47.2 Member Function Documentation

5.47.2.1 get()

```
def my_app.views.control_objective_generate_view.GenerateForControlView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.47.2.2 post()

```
def my_app.views.control_objective_generate_view.GenerateForControlView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after control objectives creation.

5.48 my_app.views.home_page_view.HomePageView Class Reference

Inherits TemplateView.

Public Member Functions

- def get (self, request, args, kwargs)

Static Public Attributes

- string **template_name** = "index.html"

5.48.1 Detailed Description

Class-based view for homepage.

5.48.2 Member Function Documentation

5.48.2.1 get()

```
def my_app.views.home_page_view.HomePageView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.49 my_register_form.MyRegistrationForm.Meta Class Reference

Static Public Attributes

- **model** = User
- tuple **fields**

5.49.1 Member Data Documentation

5.49.1.1 fields

```
tuple my_register_form.MyRegistrationForm.Meta.fields  [static]
```

Initial value:

```
=  ("first_name", "last_name", "username",
                "email", "password1", "password2")
```

5.50 my_app.models.CertificationScheme.Meta Class Reference

Static Public Attributes

- **ordering**

5.51 my_app.models.ControlComment.Meta Class Reference

Static Public Attributes

- **ordering**

5.52 my_app.models.Control.Meta Class Reference

Static Public Attributes

- **ordering**

5.53 my_app.models.ControlQuestion.Meta Class Reference

Static Public Attributes

- **ordering**
- **unique_together**

5.54 my_app.models.Metric.Meta Class Reference

Static Public Attributes

- **ordering**

5.55 my_app.models.ControlDomain.Meta Class Reference

Static Public Attributes

- **ordering**

5.56 my_app.models.SecurityAttribute.Meta Class Reference

Static Public Attributes

- **ordering**

5.57 my_app.models.ControlObjective.Meta Class Reference

Static Public Attributes

- **ordering**

5.58 my_app.models.SchemeRights.Meta Class Reference

Static Public Attributes

- **ordering**

5.59 my_app.models.SchemeComment.Meta Class Reference

Static Public Attributes

- **ordering**

5.60 my_app.models.ControlQuestionComment.Meta Class Reference

Static Public Attributes

- **ordering**

5.61 my_app.models.ControlObjectiveComment.Meta Class Reference

Static Public Attributes

- **ordering**

5.62 my_app.models.CloudServiceProvider.Meta Class Reference

Static Public Attributes

- **ordering**

5.63 my_app.models.CloudService.Meta Class Reference

Static Public Attributes

- **ordering**

5.64 my_app.search.MetricIndex.Meta Class Reference

Static Public Attributes

- **index**

5.64.1 Detailed Description

```
Meta class for metric.
```

5.65 my_app.search.SecurityAttributeIndex.Meta Class Reference

Static Public Attributes

- **index**

5.65.1 Detailed Description

```
Meta class for security attribute.
```

5.66 my_app.models.Metric Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)
- def indexing (self)
- def remove_indexing (self)

Static Public Attributes

- **metric_name**
- **max_length**
- **description**
- **expression**
- **measuring_interval**
- **author**
- **User**
- **default**
- **on_delete**
- **NEW**
- **APPROVED**
- **REJECTED**
- **STATUS_CHOICES**
- **CAN_ASSESS_CHOICES**
- **status**
- **choices**

5.66.1 Detailed Description

```
Model for metric
```

5.66.2 Member Function Documentation

5.66.2.1 indexing()

```
def my_app.models.Metric.indexing (
              self )
```

Method for indexing model from database to elastic

5.66.2.2 remove_indexing()

```
def my_app.models.Metric.remove_indexing (
              self )
```

Method for deindexing removed model from database

5.67 my_app.views.metric_approval_view.MetricApprovalView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.67.1 Detailed Description

Class for choosing which metric to approve/reject

5.67.2 Member Function Documentation

5.67.2.1 get()

```
def my_app.views.metric_approval_view.MetricApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

Method for rendering template with list of metrics

5.68 metric_change_form.MetricChangeForm Class Reference

Inherits Form.

Static Public Attributes

- **metric_name**
- **metric_description**
- **metric_expression**
- **metric_measuring_interval**

5.68.1 Detailed Description

```
Form for changing metric attributes
```

5.68.2 Member Data Documentation

5.68.2.1 metric_description

```
metric_change_form.MetricChangeForm.metric_description  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={'placeholder': _("Description of metric")}),
        label=_("Metric description"),
        required=False,
        max_length=Metric._meta.get_field("description").max_length)
```

5.68.2.2 metric_expression

```
metric_change_form.MetricChangeForm.metric_expression  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={'placeholder': _("Expression of metric")}),
        label=_("Metric expression"),
        required=False,
        max_length=Metric._meta.get_field("expression").max_length)
```

5.68.2.3 metric_measuring_interval

```
metric_change_form.MetricChangeForm.metric_measuring_interval  [static]
```

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={'placeholder': _("Metric measuring interval")}),
        label=_("Metric measuring interval"),
        required=False)
```

5.68.2.4 metric_name

```
metric_change_form.MetricChangeForm.metric_name  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={'placeholder': _("Name of metric")}),
        label=_("Metric name"),
        required=False,
        max_length=Metric._meta.get_field("metric_name").max_length)
```

5.69 my_app.views.metric_change_view.MetricChangeView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

5.69.1 Detailed Description

```
Class for changing, approve or reject metric.
```

5.69.2 Member Function Documentation

5.69.2.1 get()

```
def my_app.views.metric_change_view.MetricChangeView.get (
              self,
              request,
              args,
              kwargs )
```

Method for rendering template for chosen metric

5.69.2.2 post()

```
def my_app.views.metric_change_view.MetricChangeView.post (
              self,
              request,
              args,
              kwargs )
```

Method for handling approval/rejection
or change of metric

5.70 control_objective_automatization_form.MetricFormAfterRecommend Class Reference

Inherits Form.

Static Public Attributes

- **metric_sa_recommend**
- **metric_name_sa_recommend**
- **metric_description_sa_recommend**
- **metric_expression_sa_recommend**
- **metric_measuring_interval_sa_recommend**

5.70.1 Detailed Description

Form for metric attribute search after recommendation

5.70.2 Member Data Documentation

5.70.2.1 metric_description_sa_recommend

control_objective_automatization_form.MetricFormAfterRecommend.metric_description_sa_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        disabled=True)
```

5.70.2.2 metric_expression_sa_recommend

control_objective_automatization_form.MetricFormAfterRecommend.metric_expression_sa_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        disabled=True)
```

5.70.2.3 metric_measuring_interval_sa_recommend

control_objective_automatization_form.MetricFormAfterRecommend.metric_measuring_interval_sa_↩ recommend [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"),
        disabled=True)
```

5.70.2.4 metric_name_sa_recommend

control_objective_automatization_form.MetricFormAfterRecommend.metric_name_sa_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        disabled=True)
```

5.70.2.5 metric_sa_recommend

control_objective_automatization_form.MetricFormAfterRecommend.metric_sa_recommend [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=Metric.objects.all(),
        disabled=True,
        label=_("Security attribute metric"))
```

5.71 control_objective_automatization_form.MetricFormAfterSearch Class Reference

Inherits Form.

Static Public Attributes

- **metric_sa_search**
- **metric_name_sa_search**
- **metric_description_sa_search**
- **metric_expression_sa_search**
- **metric_measuring_interval_sa_search**

5.71.1 Detailed Description

Form for metric attribute search

5.71.2 Member Data Documentation

5.71.2.1 metric_description_sa_search

control_objective_automatization_form.MetricFormAfterSearch.metric_description_sa_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        disabled=True)
```

5.71.2.2 metric_expression_sa_search

control_objective_automatization_form.MetricFormAfterSearch.metric_expression_sa_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        disabled=True)
```

5.71.2.3 metric_measuring_interval_sa_search

control_objective_automatization_form.MetricFormAfterSearch.metric_measuring_interval_sa_↩
search [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"),
        disabled=True)
```

5.71.2.4 metric_name_sa_search

control_objective_automatization_form.MetricFormAfterSearch.metric_name_sa_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        disabled=True)
```

5.71.2.5 metric_sa_search

control_objective_automatization_form.MetricFormAfterSearch.metric_sa_search [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=Metric.objects.all(),
        disabled=True,
        label=_("Security attribute metric"))
```

5.72 control_objective_automatization_form.MetricGenerateForm Class Reference

Inherits Form.

Static Public Attributes

- **metric_name_generate**
- **metric_description_generate**
- **metric_expression_generate**
- **metric_measuring_interval_generate**

5.72.1 Detailed Description

```
Form for generated metric
```

5.72.2 Member Data Documentation

5.72.2.1 metric_description_generate

```
control_objective_automatization_form.MetricGenerateForm.metric_description_generate  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        max_length=Metric._meta.get_field("description").max_length)
```

5.72.2.2 metric_expression_generate

```
control_objective_automatization_form.MetricGenerateForm.metric_expression_generate  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        max_length=Metric._meta.get_field("expression").max_length)
```

5.72.2.3 metric_measuring_interval_generate

control_objective_automatization_form.MetricGenerateForm.metric_measuring_interval_generate [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"))
```

5.72.2.4 metric_name_generate

control_objective_automatization_form.MetricGenerateForm.metric_name_generate [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        max_length=Metric._meta.get_field("metric_name").max_length)
```

5.73 my_app.search.MetricIndex Class Reference

Inherits DocType.

Classes

- class Meta

Static Public Attributes

- **metric_name**
- **description**
- **expression**
- **measuring_interval**

5.73.1 Detailed Description

Class defining metric index.

5.74 control_objective_automatization_form.MetricRecommendForm Class Reference

Inherits Form.

Static Public Attributes

- **metric_recommend**
- **metric_name_recommend**
- **metric_description_recommend**
- **metric_expression_recommend**
- **metric_measuring_interval_recommend**

5.74.1 Detailed Description

```
Form for recommended metric
```

5.74.2 Member Data Documentation

5.74.2.1 metric_description_recommend

```
control_objective_automatization_form.MetricRecommendForm.metric_description_recommend  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        disabled=True)
```

5.74.2.2 metric_expression_recommend

```
control_objective_automatization_form.MetricRecommendForm.metric_expression_recommend  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        disabled=True)
```

5.74.2.3 metric_measuring_interval_recommend

control_objective_automatization_form.MetricRecommendForm.metric_measuring_interval_recommend [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"),
        disabled=True)
```

5.74.2.4 metric_name_recommend

control_objective_automatization_form.MetricRecommendForm.metric_name_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        disabled=True)
```

5.74.2.5 metric_recommend

control_objective_automatization_form.MetricRecommendForm.metric_recommend [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=Metric.objects.all(),
        label=_("Security attribute metric"))
```

5.75 control_objective_automatization_form.MetricSearchForm Class Reference

Inherits Form.

Static Public Attributes

- **find_metric**
- **metric_search**
- **metric_name_search**
- **metric_description_search**
- **metric_expression_search**
- **metric_measuring_interval_search**

5.75.1 Detailed Description

Form for metric search

5.75.2 Member Data Documentation

5.75.2.1 find_metric

control_objective_automatization_form.MetricSearchForm.find_metric [static]

Initial value:

```
=  forms.CharField(
        required=False,
        widget=forms.TextInput(
            attrs={"placeholder": _("Search metric..."), "size": 40}),
        label=_("Metric Search"))
```

5.75.2.2 metric_description_search

control_objective_automatization_form.MetricSearchForm.metric_description_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of metric")}),
        label=_("Metric description"),
        disabled=True)
```

5.75.2.3 metric_expression_search

control_objective_automatization_form.MetricSearchForm.metric_expression_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Expression of metric")}),
        label=_("Metric expression"),
        disabled=True)
```

5.75.2.4 metric_measuring_interval_search

control_objective_automatization_form.MetricSearchForm.metric_measuring_interval_search [static]

Initial value:

```
=  forms.IntegerField(
        widget=forms.NumberInput(
            attrs={"placeholder": _("Metric measuring interval")}),
        label=_("Metric measuring interval"),
        disabled=True)
```

5.75.2.5 metric_name_search

control_objective_automatization_form.MetricSearchForm.metric_name_search [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Name of metric")}),
        label=_("Metric name"),
        disabled=True)
```

5.75.2.6 metric_search

control_objective_automatization_form.MetricSearchForm.metric_search [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=Metric.objects.all(),
        label=_("Security attribute metric"))
```

5.76 my_app.apps.MyAppConfig Class Reference

Inherits AppConfig.

Public Member Functions

- def **ready** (self)

Static Public Attributes

- **name**

5.76.1 Detailed Description

```
Class for configuration
```

5.77 my_login_form.MyLoginForm Class Reference

Inherits AuthenticationForm.

Public Member Functions

- def **__init__** (self, request, args, kwargs)

Static Public Attributes

- **remember_me**

5.77.1 Detailed Description

```
Form for user login.
```

5.77.2 Member Data Documentation

5.77.2.1 remember_me

```
my_login_form.MyLoginForm.remember_me  [static]
```

Initial value:

```
=  forms.BooleanField(
        required=False,
        widget=forms.CheckboxInput(),
        label=_("Remember me"))
```

5.78 my_app.views.my_login_view.MyLoginView Class Reference

Inherits LoginView.

Public Member Functions

- def form_valid (self, form)

Static Public Attributes

- **form_class** = my_login_form.MyLoginForm

5.78.1 Detailed Description

```
Class-based view for login.
```

5.78.2 Member Function Documentation

5.78.2.1 form_valid()

```
def my_app.views.my_login_view.MyLoginView.form_valid (
              self,
              form )
```

```
Security check complete.
Log the user in and set expiry if not checked.
```

5.79 my_register_form.MyRegistrationForm Class Reference

Inherits UserCreationForm.

Classes

- class Meta

Public Member Functions

- def **__init__** (self, args, kwargs)
- def **save** (self, commit=True)

Static Public Attributes

- **email** = forms.EmailField(required=True)
- **first_name**
- **last_name**

5.79.1 Detailed Description

```
Form for registration
```

5.79.2 Member Data Documentation

5.79.2.1 first_name

my_register_form.MyRegistrationForm.first_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("First name")}),
        max_length=50,
        required=True,
        label=_("First name"))
```

5.79.2.2 last_name

my_register_form.MyRegistrationForm.last_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Last name")}),
        max_length=50,
        required=True,
        label=_("Last name"))
```

5.80 my_app.views.my_registration_view.MyRegistrationView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

5.80.1 Detailed Description

Class-based view for registration.

5.80.2 Member Function Documentation

5.80.2.1 get()

```
def my_app.views.my_registration_view.MyRegistrationView.get (
              self,
              request,
              args,
              kwargs )
```

Method for rendering template for registration

5.80.2.2 post()

```
def my_app.views.my_registration_view.MyRegistrationView.post (
              self,
              request,
              args,
              kwargs )
```

Method for handling registration

5.81 my_app.nlp.NLP Class Reference

Public Member Functions

- def __init__ (self)
- def get_nlp (self)

Static Public Member Functions

- def get_instance ()

5.81.1 Detailed Description

Singleton class for StanfordCoreNLP.

5.81.2 Constructor & Destructor Documentation

5.81.2.1 __init__()

```
def my_app.nlp.NLP.__init__ (
              self )
```

Initialisation method to initialize StanfordCoreNLP.

5.81.3 Member Function Documentation

5.81.3.1 get_instance()

```
def my_app.nlp.NLP.get_instance ( )  [static]
```

Method to get instance of the singleton (NLP).

:return: Instance of the NLP

5.81.3.2 get_nlp()

```
def my_app.nlp.NLP.get_nlp (
              self )
```

Method to get StanfordCoreNLP instance.

:return: Instance of the StanfordCoreNLP

5.82 my_app.nlp.Node Class Reference

Inherits object.

Public Member Functions

- def __init__ (self, label, parent)

Public Attributes

- **label**
- **parent**
- **children**
- **paths**
- **current_path**

5.82.1 Detailed Description

Node class for tree representation.

5.82.2 Constructor & Destructor Documentation

5.82.2.1 __init__()

```
def my_app.nlp.Node.__init__ (
              self,
              label,
              parent )
```

```
Initialisation method.
Variables from the stanford tree:
    label - type of the node
    parent - parent node
    children - list of children nodes
Variables for tree traversal:
    paths - list of paths, path contains list of children (list of lists)
    current_path - index to indicate which path to take from children_paths

:param label: Initial type of the node
:param parent: Parent of the node
```

5.83 password_reset_form.PasswordResetForm Class Reference

Inherits Form.

Static Public Attributes

- **email_or_username**

5.83.1 Detailed Description

```
Form for reset password.
```

5.83.2 Member Data Documentation

5.83.2.1 email_or_username

```
password_reset_form.PasswordResetForm.email_or_username  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": ""}),
        label=_("Email Or Username:"), max_length=254)
```

5.84 my_app.views.password_reset_view.PasswordResetView Class Reference

Inherits FormView.

Public Member Functions

- def post (self, request, args, kwargs)

Static Public Member Functions

- def validate_email_address (email)
- def send_email_to_user (user, request)

Static Public Attributes

- string **template_name** = "registration/password_reset.html"
- **form_class** = password_reset_form.PasswordResetForm

5.84.1 Detailed Description

```
Class-based view for password reset.
```

5.84.2 Member Function Documentation

5.84.2.1 post()

```
def my_app.views.password_reset_view.PasswordResetView.post (
              self,
              request,
              args,
              kwargs )
```

```
A normal post request which takes input
from field "email_or_username" (in PasswordResetForm).
```

5.84.2.2 send_email_to_user()

```
def my_app.views.password_reset_view.PasswordResetView.send_email_to_user (
              user,
              request )  [static]
```

```
This method creates context and sends email to the user.
```

5.84.2.3 validate_email_address()

```
def my_app.views.password_reset_view.PasswordResetView.validate_email_address (
              email )  [static]
```

```
This method validates if the input is an email address or not.
Its return type is boolean,
True if the input is a email address or False if its not.
```

5.85 my_app.views.rdf_export_view.RdfExportView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

5.85.1 Detailed Description

```
Class-based view for rdf export.
```

5.85.2 Member Function Documentation

5.85.2.1 get()

```
def my_app.views.rdf_export_view.RdfExportView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.85.2.2 post()

```
def my_app.views.rdf_export_view.RdfExportView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponse to export rdf.
```

5.86 my_app.views.scheme_approval_view.SchemeApprovalView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = comment_form.CommentForm

5.86.1 Detailed Description

```
Class-based view for scheme approval.
```

5.86.2 Member Function Documentation

5.86.2.1 get()

```
def my_app.views.scheme_approval_view.SchemeApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.86.2.2 post()

```
def my_app.views.scheme_approval_view.SchemeApprovalView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method. Returns HttpResponse for input request.
```

5.87 my_app.models.SchemeComment Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **scheme**
- **author**
- **User**
- **default**
- **comment_text**
- **max_length**
- **date**

5.87.1 Detailed Description

```
Model for scheme comments
```

5.88 scheme_compare_choose_form.SchemeCompareChooseForm Class Reference

Inherits Form.

Static Public Attributes

- **scheme_1**
- **scheme_2**

5.88.1 Detailed Description

```
Form for choosing schemes to compare
```

5.88.2 Member Data Documentation

5.88.2.1 scheme_1

scheme_compare_choose_form.SchemeCompareChooseForm.scheme_1 [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        queryset=CertificationScheme.objects.all(),
        empty_label=None,
        label=_("Scheme") + " 1")
```

5.88.2.2 scheme_2

scheme_compare_choose_form.SchemeCompareChooseForm.scheme_2 [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        queryset=CertificationScheme.objects.all(),
        empty_label=None,
        label=_("Scheme") + " 2")
```

5.89 my_app.views.scheme_compare_choose_view.SchemeCompareChooseView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = scheme_compare_choose_form.SchemeCompareChooseForm

5.89.1 Detailed Description

Class to manage requests for comparing schemes

5.89.2 Member Function Documentation

5.89.2.1 get()

```
def my_app.views.scheme_compare_choose_view.SchemeCompareChooseView.get (
              self,
              request,
              args,
              kwargs )
```

Method for rendering template with date for scheme edit rights

5.89.2.2 post()

```
def my_app.views.scheme_compare_choose_view.SchemeCompareChooseView.post (
              self,
              request,
              args,
              kwargs )
```

Method for managing changes done in scheme edit rights

5.90 my_app.views.scheme_compare_view.SchemeCompareView Class Reference

Inherits TemplateView.

Public Member Functions

- def get (self, request, args, kwargs)

5.90.1 Detailed Description

Class to manage requests for comparing schemes

5.90.2 Member Function Documentation

5.90.2.1 get()

```
def my_app.views.scheme_compare_view.SchemeCompareView.get (
              self,
              request,
              args,
              kwargs )
```

Method for rendering template with date for scheme edit rights

5.91 scheme_create_form.SchemeCreateForm Class Reference

Inherits Form.

Static Public Attributes

- **name**
- **identifier**
- **version**
- **publisher**

5.91.1 Detailed Description

```
Form for scheme creation.
```

5.91.2 Member Data Documentation

5.91.2.1 identifier

```
scheme_create_form.SchemeCreateForm.identifier  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": ""}),
        label=_("Identifier"),
        max_length=CertificationScheme._meta.get_field(
            "identifier").max_length)
```

5.91.2.2 name

```
scheme_create_form.SchemeCreateForm.name  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": ""}),
        label=_("Scheme's name"),
        max_length=CertificationScheme._meta.get_field(
            "scheme_name").max_length)
```

5.91.2.3 publisher

scheme_create_form.SchemeCreateForm.publisher [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}),
        label=_("Publisher"),
        max_length=CertificationScheme._meta.get_field(
            "publisher").max_length)
```

5.91.2.4 version

scheme_create_form.SchemeCreateForm.version [static]

Initial value:

```
=  forms.CharField(widget=forms.TextInput(
        attrs={"placeholder": ""}),
        label=_("Version"),
        max_length=CertificationScheme._meta.get_field(
            "version").max_length)
```

5.92 my_app.views.scheme_create_view.SchemeCreateView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = scheme_create_form.SchemeCreateForm

5.92.1 Detailed Description

Class-based view for scheme creation.

5.92.2 Member Function Documentation

5.92.2.1 get()

```
def my_app.views.scheme_create_view.SchemeCreateView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.92.2.2 post()

```
def my_app.views.scheme_create_view.SchemeCreateView.post (
              self,
              request,
              args,
              kwargs )
```

Post request method.
Returns HttpResponseRedirect after scheme creation.

5.93 my_app.views.scheme_delete_view.SchemeDeleteView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.93.1 Detailed Description

Class-based view for scheme delete.

5.93.2 Member Function Documentation

5.93.2.1 get()

```
def my_app.views.scheme_delete_view.SchemeDeleteView.get (
              self,
              request,
              args,
              kwargs )
```

Get request method. Returns HttpResponse for input request.

5.94 my_app.views.scheme_detail_view.SchemeDetailView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, kwargs)

5.94.1 Detailed Description

```
Class-based view for scheme detail.
```

5.94.2 Member Function Documentation

5.94.2.1 get()

```
def my_app.views.scheme_detail_view.SchemeDetailView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.94.2.2 post()

```
def my_app.views.scheme_detail_view.SchemeDetailView.post (
              self,
              request,
              kwargs )
```

```
Post method. In this method certification scheme is updated
as published (is_published = true) and site is refreshed.
```

5.95 my_app.views.scheme_edit_view.SchemeEditView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Attributes

- **form_class** = scheme_create_form.SchemeCreateForm

5.95.1 Detailed Description

```
Class-based view for scheme edit.
```

5.95.2 Member Function Documentation

5.95.2.1 get()

```
def my_app.views.scheme_edit_view.SchemeEditView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.95.2.2 post()

```
def my_app.views.scheme_edit_view.SchemeEditView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after scheme edit.
```

5.96 scheme_import_form.SchemeImportForm Class Reference

Inherits Form.

Static Public Attributes

- **name**
- **publisher**
- **identifier**
- **version**

5.96.1 Detailed Description

```
Form for scheme import
```

5.96.2 Member Data Documentation

5.96.2.1 identifier

```
scheme_import_form.SchemeImportForm.identifier  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _(
            "Scheme identifier...")}),
        label=_("Identifier"),
        max_length=CertificationScheme._meta.get_field(
            "identifier").max_length)
```

5.96.2.2 name

```
scheme_import_form.SchemeImportForm.name  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Scheme name...")}),
        label=_("Name"),
        max_length=CertificationScheme._meta.get_field(
            "scheme_name").max_length)
```

5.96.2.3 publisher

scheme_import_form.SchemeImportForm.publisher [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _(
            "Scheme publisher...")}),
        label=_("Publisher"),
        max_length=CertificationScheme._meta.get_field("publisher").max_length)
```

5.96.2.4 version

scheme_import_form.SchemeImportForm.version [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(attrs={"placeholder": _("Scheme version...")}),
        label=_("Version"),
        max_length=CertificationScheme._meta.get_field("version").max_length)
```

5.97 my_app.views.scheme_import_view.SchemeImportView Class Reference

Inherits View.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, kwargs)

5.97.1 Detailed Description

Class based view for Scheme Import

5.97.2 Member Function Documentation

5.97.2.1 get()

```
def my_app.views.scheme_import_view.SchemeImportView.get (
              self,
              request,
              args,
              kwargs )
```

```
Method for rendering template of scheme import view
```

5.97.2.2 post()

```
def my_app.views.scheme_import_view.SchemeImportView.post (
              self,
              request,
              kwargs )
```

```
Method for opening excel file handling file opening,
 columns names, importing excel into database
```

5.98 my_app.views.scheme_overview_view.SchemeOverviewView Class Reference

Inherits TemplateView.

Public Member Functions

- def get (self, request, args, kwargs)

5.98.1 Detailed Description

```
Class-based view for scheme overview.
```

5.98.2 Member Function Documentation

5.98.2.1 get()

```
def my_app.views.scheme_overview_view.SchemeOverviewView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.99 my_app.models.SchemeRights Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)

Static Public Attributes

- **scheme**
- **CertificationScheme**
- **on_delete**
- **user**
- **User**
- **NO_RIGHTS**
- **OWNER**
- **REVIEWER**
- **EDITOR**
- **STATUS_CHOICES**
- **status**
- **choices**
- **default**
- **max_length**

5.99.1 Detailed Description

```
Model for scheme edit right
```

5.100 scheme_rights_set_form.SchemeRightsSetForm Class Reference

Inherits Form.

Public Member Functions

- def **__init__** (self, args, kwargs)

5.100.1 Detailed Description

```
Form for changing scheme edit rights.
```

5.101 my_app.views.scheme_rights_set_view.SchemeRightsSetView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

Static Public Member Functions

- def set_rights (rights, new_rights)

Static Public Attributes

- **form_class** = scheme_rights_set_form.SchemeRightsSetForm

5.101.1 Detailed Description

```
Class-based view to manage scheme edit rights.
```

5.101.2 Member Function Documentation

5.101.2.1 get()

```
def my_app.views.scheme_rights_set_view.SchemeRightsSetView.get (
              self,
              request,
              args,
              kwargs )
```

```
Get request method. Returns HttpResponse for input request.
```

5.101.2.2 post()

```
def my_app.views.scheme_rights_set_view.SchemeRightsSetView.post (
              self,
              request,
              args,
              kwargs )
```

```
Post request method.
Returns HttpResponseRedirect after changing scheme edit rights.
```

5.101.2.3 set_rights()

```
def my_app.views.scheme_rights_set_view.SchemeRightsSetView.set_rights (
              rights,
              new_rights )  [static]
```

```
This method sets rights according to new_rights.
1 => is_reviewer
2 => can_edit
3 => no_rights
```

5.102 my_app.views.security_approval_view.SecurityApprovalView Class Reference

Inherits DetailView.

Public Member Functions

- def get (self, request, args, kwargs)

5.102.1 Detailed Description

```
Class for choosing which security attribute to approve/reject
```

5.102.2 Member Function Documentation

5.102.2.1 get()

```
def my_app.views.security_approval_view.SecurityApprovalView.get (
              self,
              request,
              args,
              kwargs )
```

```
Method for rendering template with list of security attributes
```

5.103 my_app.models.SecurityAttribute Class Reference

Inherits Model.

Classes

- class Meta

Public Member Functions

- def **__str__** (self)
- def indexing (self)
- def remove_indexing (self)

Static Public Attributes

- **security_attribute_name**
- **max_length**
- **description**
- **metric**
- **Metric**
- **on_delete**
- **control_domain**
- **ControlDomain**
- **author**
- **User**
- **default**
- **NEW**
- **APPROVED**
- **REJECTED**
- **STATUS_CHOICES**
- **CAN_ASSESS_CHOICES**
- **CAN_REVIEW_CO_CHOICES**
- **status**
- **choices**

5.103.1 Detailed Description

```
Model for security attribute
```

5.103.2 Member Function Documentation

5.103.2.1 indexing()

```
def my_app.models.SecurityAttribute.indexing (
              self )
```

```
Method for indexing model from database to elastic
```

5.103.2.2 remove_indexing()

```
def my_app.models.SecurityAttribute.remove_indexing (
              self )
```

Method for deindexing removed model from database

5.104 control_objective_automatization_form.SecurityAttributeGenerateForm Class Reference

Inherits Form.

Static Public Attributes

- **security_attribute_name_generate**
- **security_attribute_description_generate**

5.104.1 Detailed Description

Form for security attribute search

5.104.2 Member Data Documentation

5.104.2.1 security_attribute_description_generate

```
control_objective_automatization_form.SecurityAttributeGenerateForm.security_attribute_↩
description_generate  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of security attribute")}),
        label=_("Security attribute description"),
        max_length=SecurityAttribute._meta.get_field("description").max_length)
```

5.104.2.2 security_attribute_name_generate

```
control_objective_automatization_form.SecurityAttributeGenerateForm.security_attribute_name_↩
generate  [static]
```

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Name of security attribute")}),
        label=_("Security attribute name"),
        max_length=SecurityAttribute._meta.get_field(
            "security_attribute_name").max_length)
```

5.105 my_app.search.SecurityAttributeIndex Class Reference

Inherits DocType.

Classes

- class Meta

Static Public Attributes

- **security_attribute_name**
- **description**

5.105.1 Detailed Description

```
Class defining security attribute index.
```

5.106 control_objective_automatization_form.SecurityAttributeRecommendForm Class Reference

Inherits Form.

Static Public Attributes

- **security_attribute_recommend**
- **security_attribute_name_recommend**
- **security_attribute_description_recommend**

5.106.1 Detailed Description

```
Form for security attribute recommend
```

5.106.2 Member Data Documentation

5.106.2.1 security_attribute_description_recommend

control_objective_automatization_form.SecurityAttributeRecommendForm.security_attribute_↩
description_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of security attribute")}),
        label=_("Security attribute description"),
        disabled=True)
```

5.106.2.2 security_attribute_name_recommend

control_objective_automatization_form.SecurityAttributeRecommendForm.security_attribute_name↩
_recommend [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Name of security attribute")}),
        label=_("Security attribute name"),
        disabled=True)
```

5.106.2.3 security_attribute_recommend

control_objective_automatization_form.SecurityAttributeRecommendForm.security_attribute_↩
recommend [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=None,
        label=_("Recommended security attributes"))
```

5.107 control_objective_automatization_form.SecurityAttributeSearchForm Class Reference

Inherits Form.

Static Public Attributes

- **find_security_attribute**
- **security_attribute**
- **security_attribute_name**
- **security_attribute_description**

5.107.1 Detailed Description

```
Form for security attribute search
```

5.107.2 Member Data Documentation

5.107.2.1 find_security_attribute

```
control_objective_automatization_form.SecurityAttributeSearchForm.find_security_attribute
[static]
```

Initial value:

```
=  forms.CharField(
        required=False,
        widget=forms.TextInput(
            attrs={"placeholder": _("Search security attribute..."),
                   "size": 40}),
        label=_("Security Attribute Search"))
```

5.107.2.2 security_attribute

```
control_objective_automatization_form.SecurityAttributeSearchForm.security_attribute  [static]
```

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        required=False,
        queryset=SecurityAttribute.objects.all(),
        label=_("Security attribute"))
```

5.107.2.3 security_attribute_description

control_objective_automatization_form.SecurityAttributeSearchForm.security_attribute_description
[static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={"placeholder": _("Description of security attribute")}),
        label=_("Security attribute description"),
        disabled=True)
```

5.107.2.4 security_attribute_name

control_objective_automatization_form.SecurityAttributeSearchForm.security_attribute_name
[static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={"placeholder": _("Name of security attribute")}),
        label=_("Security attribute name"),
        disabled=True)
```

5.108 security_change_form.SecurityChangeForm Class Reference

Inherits Form.

Static Public Attributes

- **security_attribute_name**
- **security_attribute_description**
- **metric**
- **metric_details**

5.108.1 Detailed Description

Form for changing security attributes

5.108.2 Member Data Documentation

5.108.2.1 metric

security_change_form.SecurityChangeForm.metric [static]

Initial value:

```
=  forms.ModelChoiceField(
        widget=forms.Select(),
        queryset=Metric.objects.filter(status=Metric.APPROVED),
        label=_("Metric"),
        required=False)
```

5.108.2.2 metric_details

security_change_form.SecurityChangeForm.metric_details [static]

Initial value:

```
=  forms.BooleanField(
        label=_("Metric details"),
        widget=forms.CheckboxInput(),
        initial=False,
        required=False)
```

5.108.2.3 security_attribute_description

security_change_form.SecurityChangeForm.security_attribute_description [static]

Initial value:

```
=  forms.CharField(
        widget=forms.Textarea(
            attrs={'placeholder': _("Description of security attribute")}),
        label=_("Security attribute description"),
        required=False,
        max_length=SecurityAttribute._meta.get_field("description").max_length)
```

5.108.2.4 security_attribute_name

security_change_form.SecurityChangeForm.security_attribute_name [static]

Initial value:

```
=  forms.CharField(
        widget=forms.TextInput(
            attrs={'placeholder': _("Name of security attribute")}),
        label=_("Security attribute name"),
        required=False,
        max_length=SecurityAttribute._meta.get_field(
            "security_attribute_name").max_length)
```

5.109 my_app.views.security_change_view.SecurityChangeView Class Reference

Inherits FormView.

Public Member Functions

- def get (self, request, args, kwargs)
- def post (self, request, args, kwargs)

5.109.1 Detailed Description

```
Method for handling approval/rejection
or change of security attributes.
```

5.109.2 Member Function Documentation

5.109.2.1 get()

```
def my_app.views.security_change_view.SecurityChangeView.get (
              self,
              request,
              args,
              kwargs )
```

```
Method for rendering template for chosen metric
```

5.109.2.2 post()

```
def my_app.views.security_change_view.SecurityChangeView.post (
              self,
              request,
              args,
              kwargs )
```

```
Method for handling approval/rejection
or change of security metric
```

5.110 my_app.admin.UserCreationFormExtended Class Reference

Inherits UserCreationForm.

Public Member Functions

- def **__init__** (self, args, kwargs)

Static Public Attributes

- **site_url**

5.110.1 Detailed Description

```
Class to extend user creation form.
```