Metodika verziovania

Pre verziovanie projektu v našom tíme využívame verziovací systém Git[1] v konkrétnej aplikácií Github[2], kde sme si založili repozitár s konkrétnym usporiadaním súborov. Viac o repozitári v podkapitole Repozitár.

Základnými funkciami Gitu sú:

- git commit
- git push
- git pull
- git clone
- git checkout
- git status
- git diff
- git stash

Funkcií je oveľa viac, čo dodáva jazyku veľmi vysokú kvalitu a prispôsobivosť pre všetky prípady, aj v situácií, kedy sa vyskytnú problémy, no pre naše účely by mali tieto príkazy naplno stačiť (za podmienky, že sa bude dodržiavať metodika).
V nasledujúcej časti je opísaná základná funkcionalita s príkladmi, ak sú nutné.

git clone

- slúži na vytvorenie ssh spojenie (existuje aj HTTPS verzia) a vytvorí lokálnu kópiu repozitára, ktorý bude prepojený s originálom a bude sa dať pomocou ostatných git príkazov meniť
- kópia je vytvorená z defaultnej (základnej) branche

Príklad:
git clone git@github.com:FloofyReal/TP-DeepSeach.git*

*v prípade, že nepatríte do projektu alebo nemáte úspešne nastavený ssh kľúč vašeho zariadenia a Githu účtu, príkaz neprebehne správne

git pull

- slúži na aktualizovanie repozitára (pod konkrétnou branch-ov)
- stiahnu sa všetky pushnuté zmeny (commity) ostatných spolupracujúch a aplikujú sa na aktuálne súbory vo vašej lokálnej kópií repozitáru
- v prípade nálezu kolízií (potvrdenej alebo uloženej zmeny od vás a online repozitárom) bude nútení skontrolovať a upraviť dané súbory (s pomocou príkazu git diff)

[1] https://git-scm.com/
[2] https://github.com/

Príklad:
git pull*
*v prípade, že nie je nastavený rodičovský vzťah k repozitáru musíte konkretizovať z ktorej branche treba brať zmeny, tj.:
git pull origin nazov_branche
alebo prednastaviť rodičovský repozitár, z ktorého bude brať informácie a zmeny
git branch --set-upstream-to=orígin/nazov_rodicovskej_branche nazov_aktualnej_branche

git commit

- slúži na “potvrdenie” (preklad slova commit) zmien na vašej lokálnej branchi
- po commitnutí súboru ste vyzvaný napísať správu o tom, čo sa commituje (viac o tom na konci kapitoly o verziovaní)
- súbory musia byť predtým pridané príkazom: git add <názov súboru>
- commity treba robiť často, pre efektívne využitie sily verziovacích nástrojov akým je Git
- prepínač -s vás podpíše pod commit, takže sa bude dať rýchlejšie a jednoduchšie zistiť, kto daný commit vykonal

Príklad:
git add xmlparser.py
git commit -sm Add new script for parsing xml files

git push

- slúži na zosynchronizovanie vašej lokálnej branche a online branche, pričom budú vami doteraz nepushnuté commity aplikované na online branchu
- spôsob ako zverejniť vami vykonané zmeny na server

git checkout

- slúži na zmeny, obnovovanie a vytvorenie nových branchí
- git checkout <názov branche> zmení vašu aktuálnu lokálnu branchu na vami vybranú, pričom následné commity budú aplikované na ňu (vždy na aktuálnu, ak nie je inak určené)
- git checkout -b <názov novej branche> vytvorí novú lokálnu branchu, ako subbranchu vašej aktuálnej (pozor! zoznam online branchí sa ne-aktualizuje, kým danú branchu nezverejníte, bude fungovať iba lokálne)
- git checkout <názov súboru> vráti súbor do pôvodnej formy, podľa stavu v akom je v online branchi

git status

- slúži na vizualizáciu aktuálnych zmien vo vašom repozitáry (vo vašej branchi) z pohľadu súborov
- vidíte tu zmeny súborov, vytvorené nepridané súbory, vymazané neaktualizované súbory
- zmeny aktualizujete pomocou príkazov git add <názov súboru> a git commit pre uloženie zmien (vytvorenie/vymazanie súboru je tiež zmena)

git diff

- slúži na vizualizáciu aktuálnych zmien vo vašom repozitáry (vo vašej branchi) z pohľadu zmien
- vidíte tu konkrétne zmeny v konkrétnych súboroch (červené odobratie textu, zelené pridanie textu)

git stash

- uloženie aktuálneho stavu rozpracovaného repozitáru (na konkrétnej branchi) do úložiska (princíp zásobníku)
- zmeny sa dajú vyberať a naspäť aplikovať v rôznych branchách, v prípade, že pracujete na zlej a uvedomíte si to príliš neskoro
- funguje aj ako záchranný príkaz, pri vytvorení zmien, ktoré stihol od vašeho posledného pullu niekto implementovať a pushnúť, môžete si vaše zmeny stash-núť (uložiť preč), pullnúť si zmeny a daný stash vymazať, keďže je nepotrebný, miesto toho aby ste si lokálne zmeny čistili, aby pri git pulle nevznikli kolízie (alebo aby sa vám vôbec podaril vykonať príkaz git pull)

Repozitár

Repozitár má tieto hlavné zložky:

- parser
- tests
- other
- misc

V zložke **parser** sa nachádza hlavné jadro našej aplikácie, všetky skripty, ktoré sa budú využívať v priebehu vývoja, či len na pomocné vyhodnotenia alebo nevyhnutné súčasti finálnej aplikácie. Zložka parser je ďalej štrukturovaná do pod-zložiek podľa funkcionality. Tie sú zapísané na hlavnej stránke repozitára na Githube a v aktuálnom stave sú nasledujúce:

- xml - pre prácu s XML súbormi
    - discriminátor - pre indetifikáciu elementov v XML súboroch
    - article - pre spájanie konkrétnych elementov do logických článkov z XML súboru

V zložke **tests** sa nachádzajú testy pre funkcie a skripty zo zložky scripts. Budú organizované identicky (zrkadlovo) oproti zložke scripts, aby k nim viedla rovnaká cesta (odmysliac si zložku scripts/tests). Po ukončení väčšieho bloku testov ich budeme spájať a spúšťať pomocou shell scriptu všetky naraz, ktorý následne budeme môcť využiť pri aplikácií CI (continuous integration) do nášho projektu. To nám umožní efektívne testovať celú aplikáciu a šetriť čas pre následný vývoj.
V zložke **other** sa nachádzajú trénovacie a inak neoznačené skripty, ktoré chceme zachovať pre zmysel zálohovania zamietnutých prístupov (pre lepšiu dokumentáciu). Ako príklad by som uviedol spracovávanie XML súborov, pre ktoré existuje viacero využívaných knižníc,

pričom v prípade rozpracovania dvoch je rozumné si udržať prácu na tej, ktorá nebola pre projekt vybraná v prípade, že sa nám v budúcnosti ponúkne možnosť ju využiť. Táto zložka nie je nevyhnutou súčasťou projektu a slúži hlavne pre developerov (to znamená, že nepatrí pod výsledný produkt). Tak isto sa tu nachádzajú úlohy pre trénovanie členov tímu, pre zkonzistentnenie našich skúseností vo všetkých nevyhnutných technológiách.
V zložke **misc** sa nachádzajú textové súbory so zoznamami knižníc, ktoré sa používajú pri inštalácií virtuálnych prostredí. Dva hlavné súbory sú core-requirements.txt a test-requirements.txt .

Verziovanie - branche

Pre efektívne spracovanie našich úloh sme sa rozhodli pre systém, kedy máme 2 základné, chránené branche - master a development. To že sú branche chránené znamená, že sa nedá meniť priamo, ale iba pomocou pull requestov (po code review). Brancha master reprezentuje aktuálnu verziu nášho nástroja po konci šprintu a development reprezentuje najaktuálnejšiu verziu s aktuálnymi zmenami v priebehu šprintu.

Následne pri rozdelení si úloh si každý člen tímu rozdelí svoju úlohu na logické celky (stačí aj 1), podľa ktorého si vytvorí branchu s vhodným pomenovaním, na ktorej bude pracovať a následne sa vytvorí pull request a rieši sa code review (viac v ďalšej kapitole).

Mená branchí

Vzhľadom na rozmanitosť kódu a úloh sme sa rozhodli definovať hlavné mená skupín branchí pre ľahšie sa orientovanie v ich význame bez nevyhnutného kontrolovania úloh v ScrumDesku.
Základné názvy sú:

- feature - nová funkcionalita alebo modul
- enhc - vylepšenie už existujúcej funkcionality / modulu / celej aplikácie
- bugfix - oprava chyby
- tests - pridanie testov
- docs - pridanie dokumentácie

Je dôležité dodržať krátkosť a deskriptívnosť mien. V prípade, že treba dodať ďalší level komplexnosti, slová sa rozdelia pomocou podčiarkovníka (‘_’).

Verziovanie - commit správy

Pre konzistentnosť na ďalšej úrovni a to commit správy sme sa rozhodli využiť pravidlá definované viacerými zdrojmi, no prehľadne spísané a opísané v tomto blogu[3].

Pre prehľadnosť si tu definujem základné pravidlá z daného dokumentu:

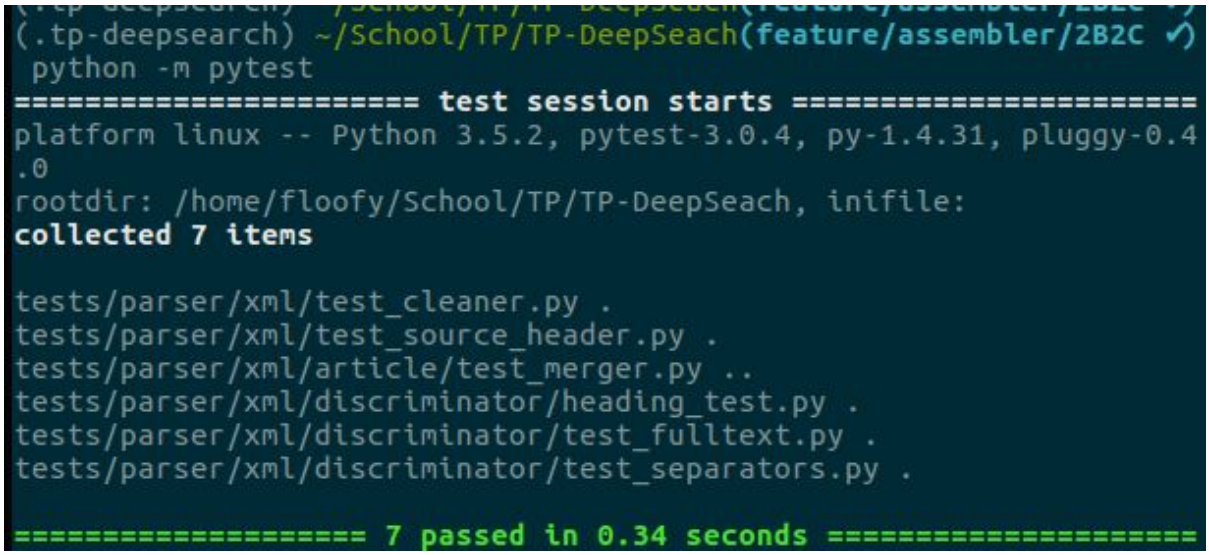
1. Oddelovať nadpis a komentár commit správy jedným prázdnym riadkom
2. Obmedz nadpis na 50 znakov
3. Prvé písmeno veľké v nadpise

[3] http://chris.beams.io/posts/git-commit/

4. Nepísať bodku na konci nadpisu
5. Používať rozkazovací spôsob nadpisu správy (imperatív)
6. Šírka tela komentáru obmedz na 72 znakov
7. V tele komentáru vysvetli Čo? a Prečo? na rozdiel od Ako?

Metodika písania testov a testovania

Pri pridaní funkcionality do programu je treba k nemu napísať reprezentatívny test do zložky test podľa štruktúry uvedenej pri definícií repozitára. Pre testovanie využívame modul Pytest, ako veľmi kvalitný nástroj pre testovanie v jazyku Python.
Daný test musí mať názov test_<definičný_názov_test>.py a v konzole sa spúšťa pomocou Pytest modulu:
python -m pytest <cesta k testu>
V prípade vynechania cesty, Pytest vyhľadá všetky súbory s názvom test a všetky metódy / funkcie s názvom test a vykoná ich, pričom farebne rozlíši úspešné alebo neúspešné vykonanie testov aj s podrobnou správou o chybe (v prípade neúspechu).



Obr. Zbehnutie testov v konzole (screenshot - stav po 4. šprinte)

Do budúcna máme v pláne implementovať nástroj pre Kontinuálnu integráciu (CI), ktorý bude kontrolovať testy automaticky a odľahčí sa teda práca vývojára, keďže sa mu automaticky ukáže, ktoré testy úspešne zbehli a ktoré nie.

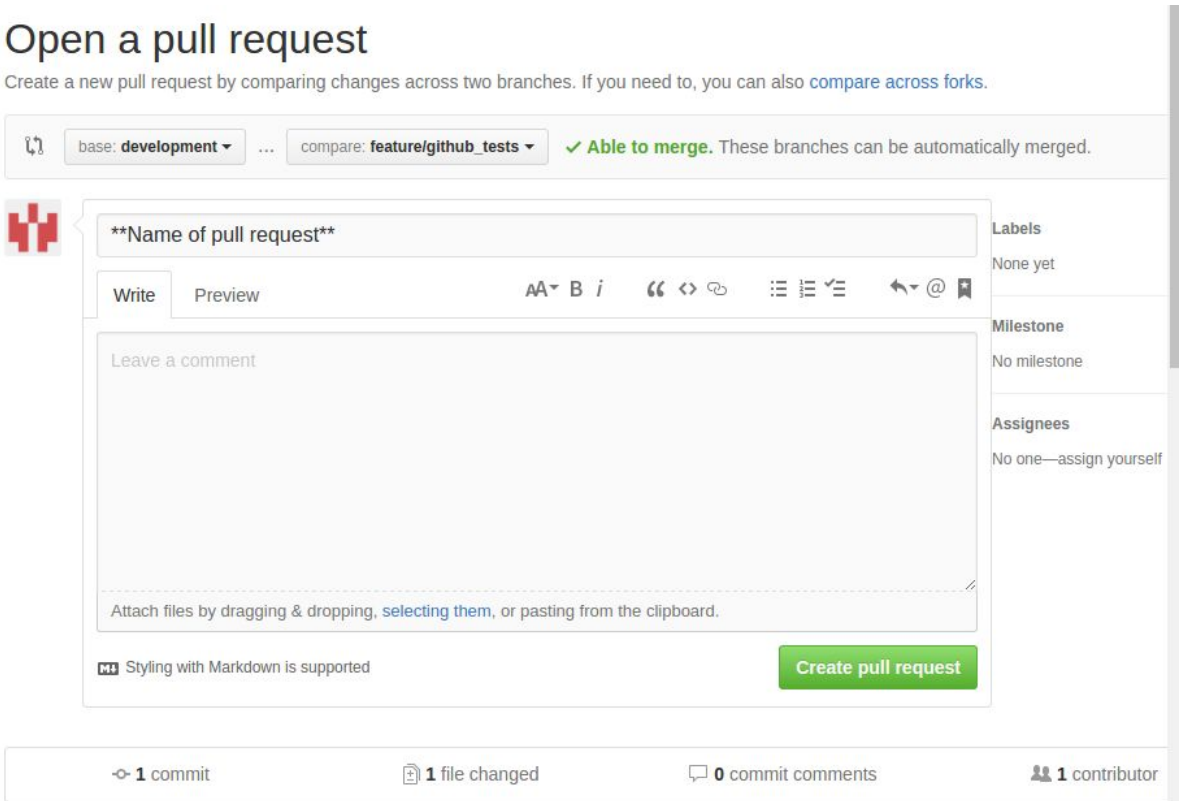
Metodika code review

Code review je nevyhnutnou súčasťou vývoja akejkoľvek aplikácie, či pre overenie správnosti, dodržanie kvality kódu alebo len ako uistenia sa ďalším párom očí, že zmena, ktorá bude vykonaná je vhodná a korektná.
Keďže ako hlavný verziovací nástroj využívame aplikáciu Github, rozhodli sme sa využívať ich implemetáciu code review pomocou pull requestov.
Pull request reprezentuje žiadosť o spojenie dvoch branchí. Následne vysvetlím princípy nášho code review na príklade.

Predpoklad: Mám vytvorenú vlastnú branchu z development branche, kde som pracoval na úlohe pridania novej funkcionality. V mojej branchi mám všetky zmeny označené a popridávané vhodnými commit správami a pushnuté, aby sa online verzia zhodovala s mojou lokálnou. Kód dodržuje kvalitu podľa štýlu PEP8 a v ideálnom prípade, sú preň napísané aj testy, ktoré všetky úspešne zbehli.

Kroky:

1. Idem na stránku našeho repozitára na Githube a nájdem tlačidlo New pull request.
2. Vyberiem si branchu, do ktorej chcem pridať moje zmeny (s ktorou chcem spojiť moju branchu) ako “base” a moju branchu ako “compare”



3. Po vybraní vhodného názvu (reprezentácia zmien, ktoré chcem spojiť) a prípadného komentára, ak je nutný, vytvorým pull request.

4. Následne, vývový líder tímu (alebo iný z developerov - vysvetlené nižšie) si otvorí daný pull request a uvidí, aké commity boli vykonané, aké sú zmeny na súboroch (github ponúka kvalitnú vizualizáciu zmien aká sa poskytuje pri príkaze git diff, bez toho aby si daný kód sťahoval a kontroloval sám)
5. Za predpokladu, že sú všetky vyššie uvedené predpoklady splnené, skontroluje logickú následnosť a má možnosť daný pull request:
    a. schváliť - v prípade, že nemá žiaden komentár a je spokojný s prácou
    b. schváliť s komentárom - väčšinou sa jedná o drobné poznámky, ktoré je dobré aby boli zdokumentované alebo nechce zbytočne otvárať ďalší komunikačný tok medzi ním a developerom
    c. zamietnuť s komentárom - objaví PEP8 chybu, logickú nezrovnalosť alebo napíše inú otázku, na ktorú treba odpoveď / vysvetlenie, alebo rovno opravenie kódu pred jeho spojením
6. Za malé pull requesty stačí zodpovednosť iného developera pre spojenie kódu, v prípade, že sa jedná o veľký pull request, je treba schválenie lídra vývoja alebo ďalšieho člena tímu (aspoň 2ja developeri musia schváliť zmenu), pričom komunikácia je nutná vopred, aby sa nepremárnilo priveľa času

Metodika code quality

Pri programovaní chceme zaručiť vysokú čitateľnosť a konzistentnosť kódu medzi rôznymi developermi. To sa nám podarí iba v prípade zadelenia veľmi konkrétnych pravidiel na najlepšie všetky prípady, aké môžu nastať, čo sa týka písania kódu. Miesto toho aby sme si vymýšľali a spisovali vlastné pravidlá, rozhodli sme sa aplikovať pravidlá pre písanie kódu overené nie len ľudmi z praxe ale podporované aj online komunitami, čo nám slúži ako dôkaz kvality tohto dokumentu. Keďže náš projekt je primárne orientovaný na jazyk Python, využívame metodiky s názvom PEP8[4]. Táto metodika, konkrétne štýl písanie Python kódu je veľmi rozsiahla a spĺňa všetky naše doterajšie potreby a tak isto má podporu v rôznych

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

vývojových prostrediach (napríklad PyCharm od JetBrains, ktorý niektorí z členov tímu využívajú) alebo aj ako konzolová aplikácia (flake8) pre kontrolovanie správnosti kódu.

V prípade, že by sa vyskytla situácia, kedy sa budeme zapodievať s otázkami štýlu, ktorý nie je zadefinovaný v PEP8, v ďalšej časti ho dôkladne zdokumentujeme, aby bolo jasné, podľa čoho sme sa orientovali.

Metodika komunikácie

Táto metodika je určená na definovanie postupov ako komunikovať v tíme. Nachádzajú sa tu informácie o rôznych nástrojoch pre rôzne typy komunikácie.

Komunikačné nástroje

Slack

Slack je primárny nástroj pre komunikáciu týkajúcu sa riešenia úloh, stretnutí a zodpovedanie všeobecných otázok. Pre každú rôznu komunikáciu je určený kanál, v ktorom sa daná problematika rieši.

Pre riešenie novej problematiky je potrebné vytvoriť vždy nový kanál a oznámiť tento kanál do #annoucements

Kanály na Slacku:
#announcements

- oznamy nových kanálov
- oznamy ohľad dôležitých udalostí
- oznamy ohľadom pridania, zmeny v ScrumDesku

#codereview
- informácie o prebehu code review
- problémy

#github_issues
- problémy v kóde
- riešenie problémov

#server
- veci týkajúce sa serveru
- webová stránka

#standup_meetings
- plánovanie stretnutí
- poznámky zo stretnutí

#tests
- vytváranie testov
- problémy s testovaním

#user_stories
- informácie k úlohám
- pomoc s vypracovaním úloh
- všetko spojené s User Stories

ScrumDesk

V ScrumDesku prebieha komunikácia týkajúca sa UserStories a Taskov. Komunikácia prebieha dvoma spôsobmi:

- Comments - komentáre k úlohe, kde sa pridávajú problémy a poznatky získané pri riešení úloh.
- Worklog - vo forme dokumentov, v ktorých sú uvedené postupy, kódy, ktoré môžu pomôcť ostatným členov tímu pri práci na úlohe.

TeamSpeak

Primárny nástroj pre hlasovú komunikáciu, prevažne pri standup meetingoch. TeamSpeak[5] nahradil neúspešne zvolené nástroje Appear.in[6], Skype[7] a Discord[8].

[5] http://www.teamspeak.com/
[6] https://appear.in/
[7] https://www.skype.com/en/

E-mail

Mailová komunikácia slúži len na kontakt s verejnosťou, v tímovej komunikácií nehrá rolu. K mailovej schránke má prístup každý člen tímu.

Riešenie problémov

1. Vytvorím kanál (ak ešte neexistuje) pre úlohu, s ktorou je problém
2. Napíšem, aký mám problém
3. Ak do 30 minút nikto nereaguje, píšem správu vedúcemu tímu, prípadne členovi tímu, ktorý je UserStory, v ktorom je problémová úloha, zodpovedný
4. Ak do 30 minút nikto nereaguje, volám na telefón vedúcemu tímu, prípadne členovi tímu, ktorý je UserStory, v ktorom je problémová úloha, zodpovedný
5. Ak nedvíha a nereaguje do 30 minút, volám vedúcu projektu.

Metodika organizovania výstupných dokumentov

Táto metodika je určená na definovanie postupov ako organizovať a pomenovávať dokumenty - výstupy a výsledky z práce, úloh a user stories, v zdieľanom priečinku. Zdieľaný priečinok má 3 podpriečinky: Worklog, Tasks, UserStories

Po skončení práce - Worklog

Po vykonaní upráce, ktorej výsledkom je dokument, sa tento dokument zaeviduje pridaním do podpriečinku Worklog. Dokument nazvite týmto spôsobom:

číslo úlohy_priezvisko - napr: 254112_vasko

[8] https://discordapp.com/

Po dokončení úlohy - Tasks

Po dokončení úlohy, ktorej výsledkom je dokument, sa vytvorí sumárny dokument zo všetkých dokumentov vo worklogu patriacich k danej úlohe. Sumárny dokument vytvára člen tímu zodpovedný za danú úlohu. Tento dokument sa zaeviduje pridaním do podpriečinku Tasks. Dokument nazvite týmto spôsobom:

číslo user story_číslo úlohy - napr: 144733_283600

Po dokončení user story - UserStories

Po dokončení user story, ktorej výsledkom je dokument, sa vytvorí sumárny dokument zo všetkých dokumentov, ktoré sú výstupmi úloh, patriacich k danej user story. Sumárny dokument vytvára člen tímu zodpovedný za danú user story. Tento dokument sa zaeviduje pridaním do podpriečinku UserStories. Dokument nazvite týmto spôsobom:

číslo šprintu_číslo user story_názov user story - napr: 2_122986_Prečistenie_a_sprehladnenie_XML_súborov (ak názov presahuje maximálne povolenú dĺžku názvu dokumentu, nahraďte názov kľúčovými slovami)

Forma výstupného dokumentu user story:

Pri vytváraní súhrnného dokumentu pre danú user story sa musia dodržiavať tieto pravidlá:

- Dokument musí obsahovať vygenerovaný obsah
- Nadpis 1. úrovne - názov user story
- Nadpisy 2.úrovne - názvy taskov
- Nadpisy 3. úrovne - iné názvy v dokumente

Metodika písania zápisnice

Táto metodika je určená na definovanie postupu pri písaní zápisnice z oficiálneho stretnutia, aby sa dosiahla konzistencia zápisníc, ale i urýchlila a uľahčila ich samotná tvorba. Zápisnice sú písané v nástroji MS Word.

Štruktúra a formát zápisnice

Pri písaní zápisnice sa postupuje podľa predvoleného formátu a štruktúry:

Zápisnica zo stretnutia č. *X*

Dátum konania: *dd.mm.yyyy*

Čas konania: *hh:mm – hh:mm*

Miesto konania: *XY*

Prítomní: *titul meno priezvisko*

titul meno priezvisko

Zapisovateľ: *titul meno priezvisko*

Neprítomní: *titul meno priezvisko (dôvod)*

Priebeh stretnutia:

Poznámka – *ak je relevantná*.

Opísaný priebeh stretnutia, logicky truktúrovaný do odsekov.

Výsledky stretnutia:

V bodoch sú uvedené veci, na ktorých sme sa dohodli:

- *Opísaná vec*.

Metodika pre záznam odpracovaného času

Táto metodika popisuje kroky pri zaznamenávaní odpracovaného času. Je určená pre každého člena tímu, ktorý má priradenú úlohu. Pre zaznamenávanie času sa používa ScrumDesk.

Postup:

1. Pri vytváraní úlohy sa zapíše odhad času potrebného pre jej splnenie
2. Po každej práci sa zaznamená odpracovaný čas
    a. Ak je možné zaznamená sa aj zostavajúci čas

Metodika pre rizika pri plánovaní

Táto metodika určuje postup riešenia rizík pri plánovaní šprintov. Riadi sa ním každý člen tímu.

Postup:

1. Člen tímu identifikuje možné riziko
2. Oboznámi s ním ostatných členov tímu
3. Tím navrhne riešenie
4. Pri určovaní ohodnotenia US sa započíta aj toto riziko
5. Manažér rizík zaznamená riziko a jeho riešenie

Metodika pre vytváranie backlogu a šprintov

V tejto metodike sú opísané pravidlá pre prácu pri vytváraní Backlogu a jednotlivých šprintov. Táto metodika je určená pre všetkých členov tímu, najmä pre člena tímu zodpovedného za plánovanie a ScrumMastra

Vytváranie elementov Backlogu

Pri vytváraní elementov backlogu rozlišujeme dva typy elementov.

A) User Story (US)

- časť backlogu, ktorej hodnota má priamy prínos pre zákazníka, teda je pre zákazníka relevantný jej výstup
- v ScrumDesku označená ako *User Story*
- **musí obsahovať**:

- Názov
    - stručný, výstižný
- Popis
    - kto potrebuje danú US?
    - Čo sa v nej deje?
    - Kvôli čomu ju potrebuje?
- Akceptačné kritériá
    - Definované na základe požiadaviek zákazníka
- Odhodnotenie
    - Odhodnotenie pomocou metódy planning poker

B) Internal Story (IS)

- Časť backlogu, ktorá je prínosná pre tím, či už z pohľadu riadenia tímu alebo programovania
- v ScrumDesku označená ako *Technical,* odlíšená farebne
- **musí obsahovať:**
    - Názov
        - Stručný, výstižný
    - Akceptačné kritériá
        - Definované na základe potrieb tímu
    - Ohodnotenie
        - Ohodnotenie pomocou metódy planning poker

V rámci vybraného elementu sa pri jej výbere do šprintu vytvoria jednotlivé úlohy, ktoré je potrebné vykonať pre jej ukončenie.

Ohodnocovanie metódou Planning poker

Procesu ohodnocovania US a IS sa zúčastňujú všetci členovia tímu. Ohodnotenie každej US/IS začína úvodom, ktorý obsahuje:

- Zoznámenie sa s úlohou danej US/IS a jej akceptačnými kritériami
- Prednesenie možných rizík, ktoré môžu nastať
- Zodpovedanie otázok a nejasností

Následne je možné pristúpiť k samotnému ohodnoteniu US/IS. Toto ohodnotenie prebieha buď pomocou kartičiek alebo mobilnej aplikácie, pričom platí, že:

- Každý člen tímu hodnotí individuálne, nezávisle a zvažuje pri tom svoje zručnosti a skúsenosti a zároveň aj zručnosti a skúsenosti ostatných členov tímu
- Ohodnotenie **nepredstavuje čas v hodinách**, ale náročnosť jednotlivých US/IS, do ktorej sa však ráta aj jej časová náročnosť
- V prípade nezhody, sú vyzvaní členovia tímu s najnižším ohodnotením a s najvyšším aby zdôvodnili svoje rozhodnutie. Po diskusii sa hlasovanie opakuje
- Výsledok hlasovania **musí byť jednohlasný**

V prípade je, že je **výsledok hlasovania vyšší než hodnota 13**, je potrebné túto US alebo IS rozdeliť na menšie a tie jednotlivo ohodnotiť.

Vytváranie šprintov a výber US/IS

Účasť členov tímu na stretnutí k vytvoreniu šprintu je povinná. Pre vytvorenie nového šprintu platia nasledujúce pravidlá:

1. Pred vytvorením šprintu musí byť ukončený predchádzajúci šprint, kvôli evidencii potreby prenesenia nedokončených US alebo IS do nového šprintu.
2. Pred výberom US musí byť jasné, čo je cieľom nového šprintu
3. Všetky US a IS vložené do šprintu musia spĺňať kritériá podľa pravidiel pre vytváranie elementov backlogu
4. Pripravený šprint musí byť odsúhlasený product owner-om

Pri výbere US a IS sa zohľadňujú výsledky prechádzajúcich šprintov, pričom sa **prednostne vkladajú do šprintu US a IS, ktoré boli vytvorené na základe nedokončených US a IS v predošlom šprinte.** Následne sa pridávajú US a IS, ktoré sú potrebné pre splnenie stanoveného cieľa šprintu.

Prideľovanie zodpovednosti a práce na US/IS

Pre každú US je potrebné určiť členov tímu zodpovedných za jej ukončenie. Táto zodpovednosť však **neznamená, že je daný člen povinný vykonať danú US sám.** Výber US alebo IS je **dobrovoľný,** ak však ostane daný element nepridelený, bude **pridelený Scrummastrom niektorému z členov tímu**, s ohľadom na jeho predošlú prácu a skúsenosti. Úlohou zodpovedného člena tímu je:

- Sledovanie progresu v rámci US/IS
- Hlásenie krízových situácií manažmentu plánovania a vedúcemu tímu
- Vytvorenie záverečnej dokumentácie k US/IS

V rámci každej US alebo IS je potrebné vytvoriť jednotlivé úlohy. Súčasťou každej úlohy je:

- Názov
- Popis
- Odhadovaný čas trvania *(estimate)*
- Čas, ktorý bol už vykonaný *(spent)*
- Odhadovaný čas dokončenia *(remaining)*

V prípade potreby je možné na danú úlohu vytvoriť termín dokončenia *(due date)*. Na jednotlivých úlohách môže naraz pracovať viacero členov tímu, ktorí svoju prácu dokumentujú na základe pravidiel nachádzajúcich sa v metodike pre dokumentáciu. Zodpovedný za úlohu je vždy jeden člen, ktorý má na starosti:

- Sledovanie progresu v rámci úlohy
- Vytvorenie dokumentácie k danej úlohe
- Hlásenie krízových situácií

Táto zodpovednosť je **dobrovoľná** a jednotlivé úlohy si môžu členovia tímu vyberať. V prípade, že ostane niektorá z úloh nepridelená, **pridelí úlohu Scrummaster**. V prípade potreby je možné US alebo IS rozšíriť o ďalšie úlohy.

Zmena stavov, posúvanie Taskov a UserStories po tabuľa práce

Člen tímu - Task leader

- Člen tímu, ktorý má na starosti daný Task, je za neho zodpovedný môže v rámci tabuly práce (Kanban) pohybovať s danými Taskami v rámci týchto pravidiel:
    - ak začal vykonávať danú úlohu, je povinný presunúť Task zo stavu “To Dp” do stavu “In Progress”. Zmena stavu iným spôsobom nie je povolená

ScrumMaster

- Až keď je daný Task spravený na 100 percent, teda aj s potrebnými testami, ScrumMaster tento Task smie presunúť zo stavu “In Progress” do “Done”
- Ak sú všetky Tasky v rámci User Story v stave “To Do”, potom sa daný US vyhodnotí celý ako done
- Musia byť samozrejme splnené všetky potrebné akceptačné kritériá
- V prípade ak sa daný US nestihol spraviť celý, nechá sa takto nedokončený v danom šprinte, v ďalšom šprinte sa vytvorí rovnaký US, ale iba s úlohami, ktoré neboli dokončené
    - zároveň celý tím určí nanovo Story Pointy tohto nového US, pretože pracnosť už bude pozmenená

Metodika Stand-Up

- Stand-Up - na diaľku
    - každý Pon, Ut, Str, Štv, Ne, prostredníctvom TeamSpeak
    - 15 - 60 minút
    - zhodnotenie stavu úloh, odhad, splnenie úloh, riešenie problémov, poskytovanie pomoci pri riešení úloh
- Stand-Up - interne v rámci tímu
    - každý Utorok a Stredu v priestoroch školy
    - 1 - 2 hodiny
    - zhodnotenie stavu, problémov, poskytovanie pomoci pri riešení problémov s úlohami, revidovanie kódu, ukážka práce - kódu, šírenie rád a poznatkov

- Stand-Up - oficiálne s Product Ownerom
  - každý štvrtok v priestoroch školy
  - 3 hodiny
  - retrospektíva, prezentácie vykonaných prác počas šprintu, naplánovanie ďalšieho šprintu, definícia akceptačných kritérií