Metodika dokumentovania retrospektívy šprintu

verzia 2017-11-06

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá dokumentovania retrospektívy šprintov. Retrospektíva prebieha po každom uzatvorení šprintu. Táto metodika predpisuje obsah dokumentu retrospektívy a aj formálnu stránku dokumentu.

2. Znenie metodiky

Proces: Retrospektíva šprintu na stretnutí

- Na stretnutiach tímu, počas ktorých sa uzatvára šprint, je potrebné vykonať retrospektívu práve končiaceho šprintu
- Táto retrospektíva má formu otvorenej diskusie medzi všetkými členmi tímu a vedúcimi projektu
    - Téma diskusie je, ako prebiehal šprint, čo sa podarilo a čo sa nepodarilo. Pre každý bod v časti neúspešných vlastností sa kolektívne hľadá konkrétne a realizovateľné riešenie

Proces: Dokumentovanie retrospektívy šprintu

- Počas retrospektívy vzniká dokument zachytávajúci všetky identifikované body
    - Dokument vzniká priamo na stretnutí tak, že ho všetci vidia a môžu do neho prispieť (počas písania je napríklad zobrazený na projektore)

- Ide o kolektívne dielo vznikajúce na základe diskusie, píše ho jeden člen, ale jeho meno sa v dokumente neuvádza (keďže dokument je kolektívny)

Proces: Nastavenie štruktúry dokumentu retrospektívy šprintu

- Identifikácia šprintu (jeho poradové číslo)
- Dátum, kedy dokument vznikol
- Nedostatky
  - Ako odrážky
  - Usporiadané podľa úloh
  - K nedostatkom sa priraďujú nájdené riešenia (ako pododrážky)
- Úspechy
  - Ako odrážky
  - Usporiadané podľa úloh

Proces: Uloženie dokumentu retrospektívy a jeho neskoršia modifikácia

- Dokument sa po jeho vytvorení umiestni do tímového Google Drive priečinka (do podzložky sprint_reviews)
  - Prístup na čítanie k nemu majú všetci členovia tímu aj vedúci
  - Zmeny v týchto dokumentoch bežne nie sú povolené, môžu sa vykonávať iba vo výnimočných prípadoch, a to so súhlasom všetkých členov aj vedúceho
  - V prípade, že sa dokument zmení po ukončení stretnutia, na ktorom bol vytváraný, musí byť táto skutočnosť uvedená (spolu s dôvodom zmeny) na konci dokumentu spolu s menom člena, ktorý zmenu vykonal

Metodika dokumentovania stretnutí (zápisnice)

verzia 2017-11-06

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá pre dokumentovanie oficiálnych stretnutí (raz týždenne). Metodika stanovuje, aké časti treba dokumentovať a aký má byť obsah každej časti. Metodika tiež určuje, akú formu má mať vytvorený dokument.

2. Znenie metodiky

Proces: Vytvorenie dokumentu stretnutia (zápisnice zo stretnutia)

- Zápisnica sa vytvorí čo najskôr po každom stretnutí tímu (ešte v deň stretnutia)
- V ten istý deň sa zápisnica uverejní na stránke tímu vo formáte PDF
- Počas stretnutia sa určí člen tímu zodpovedný za vytvorenie zápisnice a člen zodpovedný za jej revíziu a umiestnenie na webovej stránke tímu
- Názov súboru je zapisnica_stretnutie_X.pdf, kde X je poradové číslo stretnutia

Proces: Nastavenie štruktúry dokumentu (zápisnice):

- Nadpis: Zápisnica stretnutia tímu EduVirtual (tím číslo 4)
- Téma stretnutia
- Dátum stretnutia
- Miesto stretnutia (zvyčajne FEI STU B402)
- Zoznam prítomných a neprítomných členov (vrátane vedúceho)
- Meno zapisovateľa zápisnice

- Program stretnutia
- Vyhodnotenie úloh z predchádzajúceho stretnutia
- Opis stretnutia
- Aktuálny stav úloh v systéme manažmentu projektu

Proces: Písanie programu stretnutia

- Program stretnutia
  - V odrážkach napísané témy, okruhy alebo konkrétne otázky, ktoré sa na stretnutí riešili
  - Každý bod musí byť pochopiteľný
    - ak nie je účel témy jasný z názvu, je potrebné ho niekoľkými vetami upresniť
  - Voliteľne je možné uviesť aj meno člena, ktorý navrhol daný bod zaradiť do programu

Proces: Písanie vyhodnotenia úloh z predchádzajúceho stretnutia

- Vyhodnotenie úloh z predchádzajúceho stretnutia - Backlog ukončeného šprintu (ak počas stretnutia šprint nekončí, uvedie sa namiesto nižšie popísanej tabuľky informácia “Počas tohto stretnutia šprint ešte prebiehal, úlohy teda neboli uzavreté. Aktuálny stav úloh a priebežné výsledky sme prezentovali vedúcim a zapísali sme pripomienky.”)
  - ide o tabuľku obsahujúcu stories alebo úlohy daného šprintu vo forme:
    - ID - identifikátor story alebo úlohy v nástroji na manažovanie projektu
    - Názov - meno story alebo úlohy
    - Členovia - na prvom mieste je uvedené meno člena, ktorý je za danú story alebo úlohu zodpovedný, za čiarkou

môžu voliteľne pokračovať mená členov, ktorí sa na riešení story / úlohy podieľajú
    - Táto bunka tabuľky musí obsahovať minimálne jedno meno: zodpovedného člena
    - V prípade, že na riešení story / úlohy sa podieľajú všetci, členovia, uvedie sa na prvom mieste meno zodpovedného člena a za čiarkou sa uvedie slovo “všetci”
- Dátum zadania - dátum, kedy bola story / úloha vytvorená
- Očakávaný dátum ukončenia - dátum, kedy je naplánované dokončenie danej story / úlohy
- Stav - zhoduje sa so stavom, ktorý je danej úlohe pridelený v nástroji na manažment projektu
    - V zátvorke za názvom stavu sú uvedené percentá vyjadrujúce časť úlohy, ktorá je dokončená
    - V prípade, že je úloha uzavretá a Product Owner potvrdil splnenie Definition of Done, je táto skutočnosť uvedená v zátvorke (DOD splnené)
    - V prípade, že je úloha uzavretá a Product Owner potvrdil jej zrušenie, je v zátvorke uvedené (Zrušená)
    - Za zátvorkou môže byť uvedená krátka poznámka
    - Ak je potrebné upresniť stav, je možné využiť poznámku pod čiarou a do nej umiestniť podrobné vysvetlenie aktuálneho stavu vrátane faktorov, ktoré tento stav zapríčinili

Proces: Písanie opisu stretnutia

- Opis stretnutia - každý bod z časti Program stretnutia sa podrobne rozpíše

- Uvádza sa sem, čo zapríčinilo potrebu riešiť tento bod, aký je očakávaný cieľ diskusie, samotný priebeh diskusie so všetkými podrobnosťami a jasne zhrnutý záver

Proces: Písanie aktuálneho stavu úloh

- Aktuálny stav úloh v systéme manažmentu projektu
  - Tu sa uvádza Sprint Backlog práve prebiehajúceho alebo novovytvoreného šprintu
  - Štruktúra tabuľky sa zhoduje s tabuľkou v časti Vyhodnotenie úloh z predchádzajúceho stretnutia, ale bol pridaný jeden stĺpec:
    - Story Points / Odhad. čas - počet story pointov predelený story / úlohe a odhadovaný čas na splnenie úlohy
      - v prípade prebiehajúceho šprintu sa ako odhadovaný čas uvádza, koľko času ešte zostáva
  - V prípade, že sa na stretnutí zmení Product Backlog, uvedie sa v tejto kapitole aj tabuľka s kompletným Product Backlogom, jej štruktúra je nasledovná:
    - ID - identifikátor story alebo úlohy v nástroji na manažovanie projektu
    - Názov - meno story alebo úlohy
    - Zodpovedný člen - člen, ktorému bola story alebo úloha priradená
      - bunka môže byť prázdna (úlohy v product backlogu nemusia byť pridelené)
    - Typ - typ story alebo úlohy uvedený v nástroji na manažovanie projektu

Metodika dokumentovania sumarizácie šprintov

verzia 2017-12-11

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá dokumentovania sumarizácie šprintov. Sumarizácia šprintu je opis tvorby šprintu, priebehu šprintu, uzavretia šprintu a zhodnotenie výsledkov vytvorených v priebehu šprintu. Tento dokument predpisuje spôsob dokumentovania tejto sumarizácie po formálnej a obsahovej stránke.

Súvisiace dokumenty a metodiky:

1. Metodika manažmentu organizácie artefaktov (Google Drive). v2017-10-18.

2. Znenie metodiky

Proces: Vytvorenie sumarizácie šprintu

- Sumarizácia šprintu sa tvorí v deň vytvorenia nového šprintu a v deň ukončenia šprintu
- Sumarizácie šprintov sa pridávajú do dokumentu riadenia projektu (v tímovom Google Drive priečinku podľa metodiky [1]) do kapitoly Sumarizácie šprintov

Proces: Nastavenie štruktúry príspevku k sumarizácii šprintu

- Sumarizácia jedného šprintu má nasledujúcu štruktúru:
    - Časti, ktoré sa tvoria po vytvorení nového šprintu:
        - identifikátor šprintu (jeho číslo)

        - tabuľka zachytávajúca Product Backlog pred vytvorením backlogu nového šprintu
            - tabuľka zachytáva ID príbehov a ich názov
        - dátum začiatku šprintu
        - tabuľka zachytávajúca backlog novovytvoreného šprintu
            - tabuľka zachytáva ID príbehov a ich názov
        - popis vybraných príbehov do šprintu
        - dôvody, prečo neboli vybrané do Šprint Backlogu ďalšie príbehy z Product Backlogu
    - Časti, ktoré sa tvoria po ukončení šprintu:
        - opis priebehu šprintu, spomenutie nových poznatkov a identifikovaných problémov
        - zhodnotenie šprintu, vyznačenie príspevkov, pomenovanie nezvládnutých úloh spolu s dôvodom ich nezvládnutia

Metodika komunikácie
(Slack a tímový mail)

verzia 2017-10-18

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá komunikácie medzi členmi tímu navzájom, medzi členmi tímu a vedúcimi a medzi tímom a ďalšími osobami. K vnútrotímovej komunikácii (vrátane komunikácie s interným vedúcim) sa používa systém Slack. Ku komunikácii s externým vedúcim a ďalšími osobami sa používa emailová komunikácia cez tímový e-mail (outlook.com).

2. Znenie metodiky

Proces: Komunikácia s Product Ownerom a externými osobami

- Pre tento druh komunikácie sa používa tímový mail
- fiit_tp1718_team04@outlook.com
- Tento mail slúži na komunikáciu s product ownerom a externými osobami
- Prístup k účtu má každý člen tímu
- Nové maily píše hlavne teamleader po dohode s ostatnými členmi tímu
    - V prípade potreby môže napísať nový mail ktokoľvek s tímu, pričom o tom informuje ostatných členov tímu čo najskôr
- Predmet novej správy má formu [TP] Téma - kde Téma je výstižný názov obsahu mailu
- Podpis na konci má formu Meno Priezvisko, tím 4, kde Meno a Priezvisko je meno a priezvisko autora mailu
- Outlook automaticky zakončuje maily pätičkou:

    Tim c. 4

    EduVirtual

Timovy projekt FIIT STU

Proces: Komunikácia medzi členmi tímu a komunikácia s vedúcim

- Pre tento druh komunikácie sa používa komunikačný systém Slack
- tim4eduvirtual.slack.com
- Slack sa používa na komunikáciu a zdieľanie dát medzi členmi tímu
- Správy adresované všetkým členom sa vkladajú do kanálu #general
- Pre komunikáciu s iba jedným členom sa používajú priame správy

Metodika manažmentu globálnych cieľov projektu

verzia 2017-12-11

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá pre tvorbu, aktualizovanie a dokumentovanie globálnych cieľov projektu. Jasne definované ciele projektu sú základný predpoklad pre úspešné dokončenie projektu, pre efektívne napredovanie a pre skvalitnenie a zjednodušenie komunikácie v rámci tímu aj mimo neho. Preto si takto dôležitá časť projektu vyžaduje vlastnú metodiku.

Súvisiace dokumenty a metodiky:

1. Metodika manažmentu organizácie artefaktov (Google Drive). v2017-10-18.
2. Metodika komunikácie (Slack a tímový mail). v2017-10-18.

2. Znenie metodiky

Proces: Vytvorenie dokumentu s globálnymi cieľmi projektu

- Dokument je uložený v tímovom Google Drive priečinku
    - Miesto uloženia dokumentu určuje metodika [1]
- Dokument je prístupný všetkým členom tímu, vedúcemu a vlastníkovi produktu po celú dobu trvania projektu
- Pri každej zmene cieľov projektu je nutné dokument ihneď aktualizovať

Proces: Aktualizácia dokumentu s globálnymi cieľmi projektu

- Každú zmenu globálnych cieľov musí schváliť vlastník produktu
- Návrh na zmenu cieľov môže predložiť každý člen tímu
- Návrh na zmenu prebieha nasledovným spôsobom:
    - Člen tímu, ktorý chce vytvoriť návrh na zmenu cieľov, vytvorí novú dočasnú verziu dokumentu s globálnymi cieľmi podľa metodiky [1]
    - Návrh na zmenu premietne do dokumentu globálnych cieľov - teda upraví dokument tak, aby zodpovedal jeho predstavám o nových cieľoch
    - Informuje celý tím a vedúceho o vytvorení návrhu na zmenu globálnych cieľov podľa metodiky [2]
    - TeamLeader informuje o tomto návrhu vlastníka produktu podľa metodiky [2]
- Prebehne diskusia medzi všetkymi členmi tímu, vedúcim tímu a vlastníkom produktu
- Ak vlastník produktu zmenu schváli, verzia s návrhom zmeny sa stáva platnou verziou dokumentu s globálnymi cieľmi projektu
    - Zmena platnej verzie sa vykoná podľa metodiky [1]

Proces: Nastavenie štruktúry dokumentu s globálnymi cieľmi projektu

- Charakteristika projektu
    - Jedna veta, o čom je projekt
    - Jeden odstavec podrobnejšie opisujúci zmysel projektu, dôvod jeho vzniku, jeho smerovanie a ciele
- Kľúčové body projektu
    - V odrážkach napísané hlavné ciele, ktoré sa snaží projekt dosiahnuť
    - Maximálne 15 bodov
- Logické členenie projektu

    - Rozdelenie projektu na menšie, relatívne nezávislé časti (ak je to možné)
    - Stručný popis každej časti
- Ciele jednotlivých častí projektu
    - Pre každú časť napísať zoznam cieľov, ktoré sa snaží daná časť dosiahnuť
- Infografika
    - Jednoduchá infografika popisujúca projekt - je určená pre komunikáciu s osobami priamo nezúčastnenými pre tvorbe projektu
- Záznam zmien v globálnych cieľoch projektu
    - Zoznam vykonaných zmien - každá zmena cieľov tu je stručne opísaná
        - Dôvod zmeny
        - Výpis, ktoré ciele sa zmenili

Metodika manažmentu organizácie artefaktov (Google Drive)

verzia 2017-10-18

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá práce so zdrojmi / artefaktmi, spôsob ich pomenovávania, ukladania a kategorizovania. K ukladaniu a zdieľaniu artefaktov sa používa priečinok v službe Google Drive, ktorý je zdieľaný so všetkými členmi tímu, s interným aj externým vedúcim. Interný vedúci, externý vedúci a členovia tímu majú prístupové práva na úpravu tohto priečinku a všetkého jeho obsahu.

2. Znenie metodiky

Proces: Uloženie artefaktu do úložiska Google Drive

- Pre zdieľanie a zálohovanie dokumentov a súborov, ktoré sa nezaraďujú do žiadneho Github repozitára, je vytvorený zdieľaný priečinok TP_share na Google Drive účte teamleadera
- Plný prístup k nemu má každý člen tímu
- Vkladať súbory môže každý člen tímu podľa potreby
    - Jednotlivé súbory je možné vkladať priamo do koreňového adresára, viaceré súvisiace súbory sa vkladajú do nového priečinka
    - Názvy súborov alebo priečinkov musia výstižne popisovať obsah
- Po vložení súboru by mal autor informovať všetkých ostatných členov tímu

Metodika manažmentu úloh (Redmine)

verzia 2017-11-09

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá manažovania úloh a používateľských príbehov. Metodika taktiež určuje pravidlá pre zaznamenávanie stavu projektu, pokroku a odpracovanom čase za každého člena. Manažment úloh vychádza z metodológie Scrum. Na manažment úloh sa používa systém Redmine.

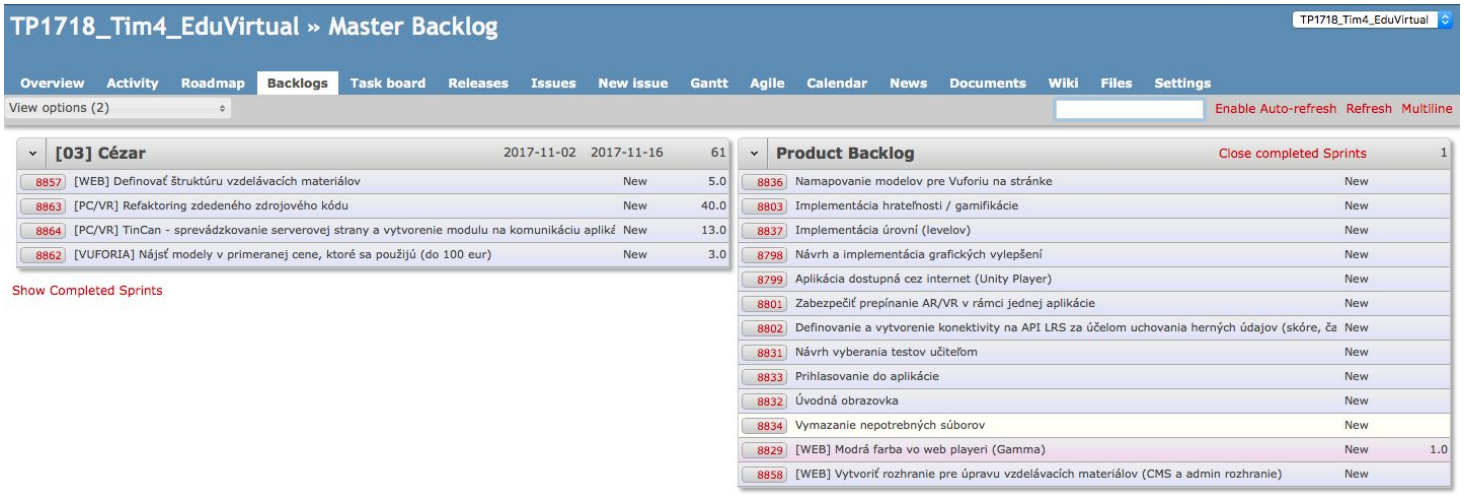
2. Znenie metodiky

Proces: Tvorba a úprava Product Backlogu

- Product Backlogobsahuje zapísané príbehy (stories)
- Tieto stories je možné deliť do vhodných epicov - väčších skupín združujúcich stories s podobným zámerom
- Každý príbeh (story) by mal mať merateľný prínos pre projekt, mal by byť dobre dokumentovateľný a mal by sa zhodovať s niektorou požiadavkou alebo očakávaním product ownera
- Product Backlog sa tvorí a upravuje na stretnutí s product ownerom
    - So systémom Redmine pracuje na stretnutí jeden člen tímu (podľa kolektívnej dohody, preferovane Scrum master), pričom jeho práca by mala byť viditeľná všetkým
    - Tím na základe konzultácie s vedúcim a product ownerom vytvorí príbehy, prípadne ich zaradí do zodpovedajúcich epicov
    - Následne sa ohodnotí zložitosť každého príbehu
        - Používa sa jednotka Story point
        - Zložitosť sa ohodnocuje technikou “pokeru” - každý člen si vyberie kartu, ktorá zodpovedá jeho odhadu zložitosti

- Všetci členovia naraz ukážu svoje karty
- Následne sa diskutuje, prečo členovia zvolili práve také odhady (hlavne minimá a maximá)
- Po ukončení diskusie sa opakuje vyberanie karty až dovtedy, kým sa všetci nezhodnú na jednej hodnote story pointov - vtedy sa táto hodnota pridelí príbehu a pokračuje sa na ďalší príbeh

- Ak je niektorý príbeh ohodnotený na 21 a viac story pointov, je vhodné zvážiť rozdelenie na menšie a jasnejšie príbehy
- Product owner usporiada príbehy podľa priority od najdôležitejšej po najmenej dôležitú
  - Ak je niektorý príbeh ohodnotený moc vysokým počtom story pointov a nevošiel by sa do najbližšieho šprintu, product owner



môže navrhnúť rozdelenie príbehu na menšie časti

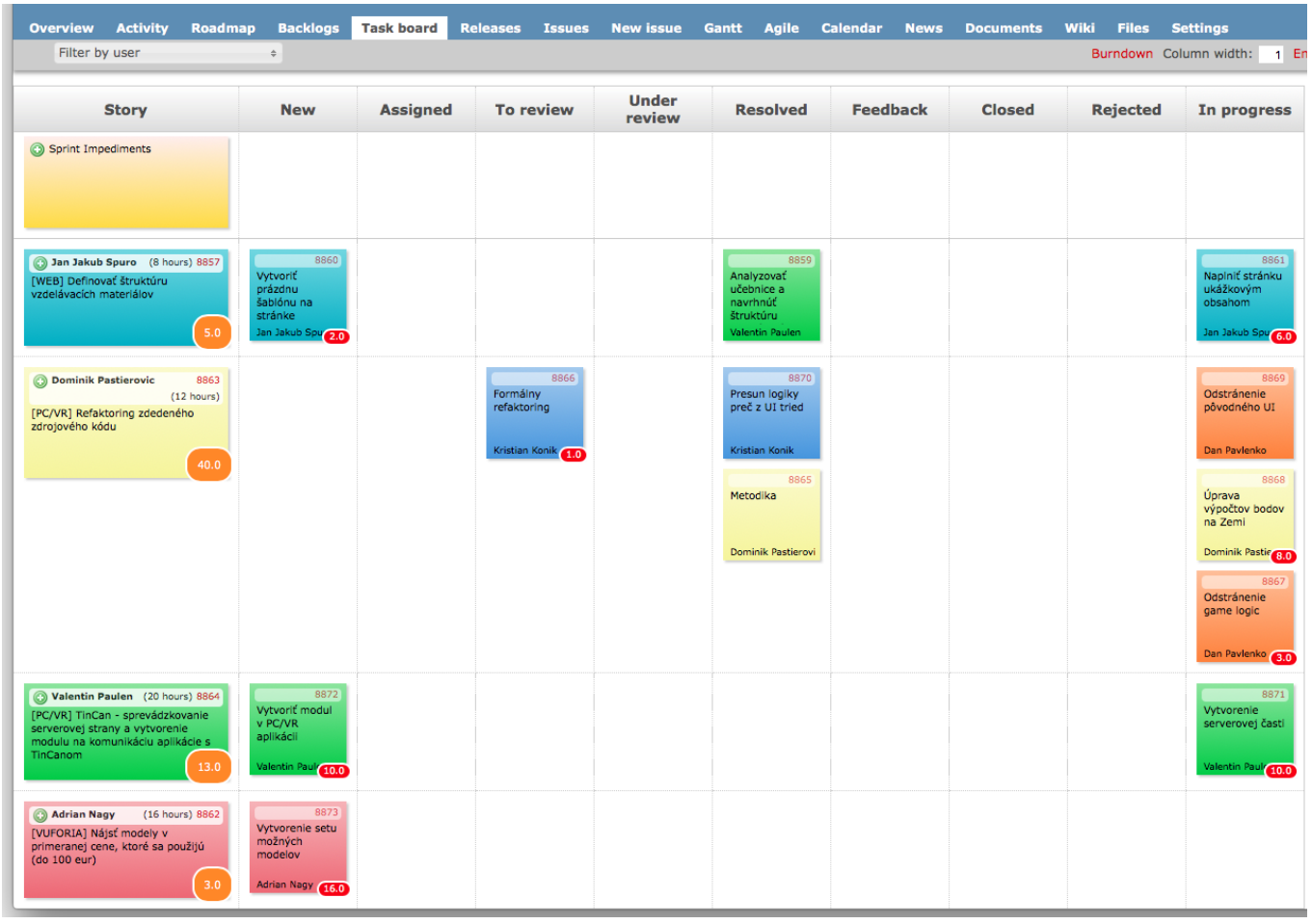
Obrázok 1: Zobrazenie príbehov v product backlogu a v šprinte

Proces: Tvorba nového šprintu

- Keď je Product Backlog usporiadaný korektne, vytvorí sa v systéme nový šprint a do neho sa presúvajú príbehy z product backlogu (obrázok 1) podľa priority tak, aby sa celkové množstvo story pointov čo najviac priblížilo určenej hranici

- Hranicu si určuje tím pri každom vytváraní šprintu podľa výsledkov minulého šprintu, šprint reviewu, prípadne celkového časového plánu na najbližšie 2 týždne
    - Vhodná je snaha postupne zvyšovať hranicu

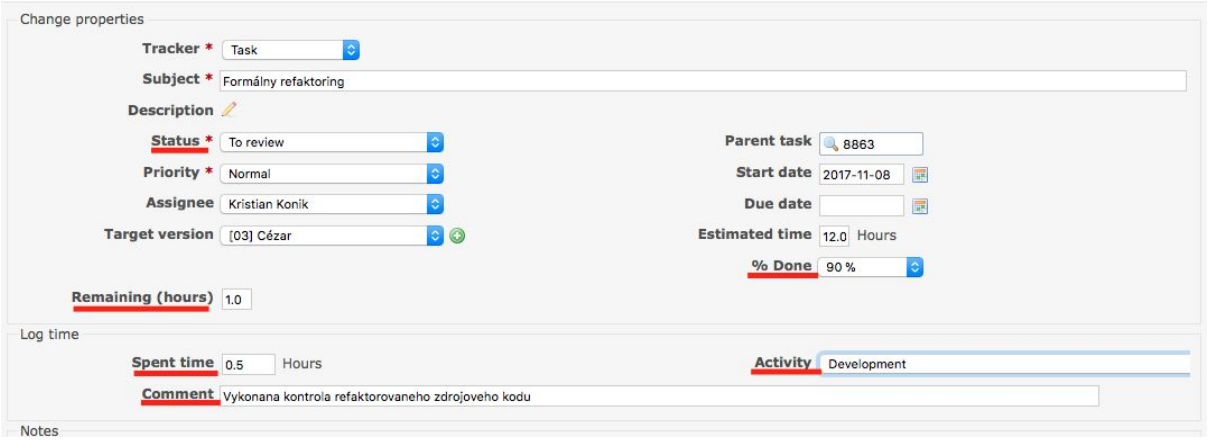
- Následne sa príbehy pridelia členom, ktorí budú za ne zodpovední, definuje sa DOD
- Príbehy sa rozdelia na úlohy, tiež sa pridelia členom tímu, definuje sa DOD a všetky podrobnosti potrebné pre vypracovanie úlohy (obrázok 2)
- Určia sa časové odhady pre každú úlohu a člen zodpovedný za vykonanie review úlohy (ak je to potrebné)
- Novovytvorené úlohy sa označia stavom New



Obrázok 2: Rozdelenie príbehov na úlohy, pridelenie úloh členom a časový odhad

Proces: Zaznamenávanie aktivity počas riešenia úlohy

- Keď člen tímu začne riešiť svoju úlohu, zmení jej stav na In progress
- Po dosiahnutí pokroku v danej úlohe si riešiteľ zaznamená čas (počet hodín, ktoré na úlohe odpracoval), typ práce (Development, Design alebo Documentation) a komentár (obrázok 3)
    - Komentár obsahuje zhrnutie vykonanej práce - stručné, ale úplné, priamočiare a pochopiteľné
- Následne riešiteľ upraví časový odhad zostávajúcej práce na úlohe a počet percent vyjadrujúci dokončenie úlohy



Obrázok 3: Zaznamenanie pokroku v úlohe

- Po dokončení úlohy zmení riešiteľ jej stav na To review
    - Osoba zodpovedná za review zhodnotí výsledok úlohy, vyznačí nedostatky a informuje o nich riešiteľa
    - Riešiteľ nastaví stav úlohy na In progress, nedostatky vyrieši a zmení stav opäť na To review
    - Tento cyklus sa opakuje, pokiaľ review neprebehne bez nedostatkov a DOD bude považované za splnené, vtedy riešiteľ úlohe nastaví stav Resolved

Proces: Uzatváranie šprintu

- Pri uzatváraní šprintu tím postupne prezentuje product ownerovi výsledky
- Ak vedúci schváli dokončenie úlohy a splnenie DOD, zmení sa stav úlohy na Closed
- Toto sa opakuje pre všetky úlohy v danom šprinte
- Príbehy, ktoré majú všetky úlohy uzavreté sa považujú za dokončené
    - Tie, ktoré neboli dokončené sa vrátia do Product Backlogu
- Ak sú všetky príbehy v Sprint backlogu dokončené (a nedokončené sú presunuté do produckt backlogu), šprint sa uzavrie
- Pokračuje sa plánovaním ďalšieho šprintu (viď. začiatok tejto kapitoly)

Metodika manažmentu verzií (Github)

verzia 2017-11-12

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá práce so systémom Github, ktorý je určený na správu verzií zdrojových kódov. Metodika obsahuje všeobecné pravidlá, ktoré je nutné dodržiavať pri práci s ktorýmkoľvek repozitárom v rámci projektu. Ďalej metodika určuje pravidlá špecifické pre každý repozitár. Tieto pravidlá je nutné dodržiavať iba pri práci s príslušným repozitárom.

V prípade, že počas práce vznikne dočasný repozitár, je možné metodiku práce s ním umiestniť do jeho wiki stránky. Ak však bude repozitár aktívne používaný viac ako 20 dní, metodika jeho používania musí byť uvedená v tomto dokumente. V prípade, že metodika použitia repozitára je uvedená v tomto dokumente aj na wiki stránke daného repozitára, je nutné riadiť sa pravidlami uvedenými v tomto dokumente (tento dokument je nadriadený wiki stránke repozitára).

2. Znenie metodiky

Proces: Commit, písanie commit správ

Pri všetkých repozitároch sa dbá na používanie zhodnej formy commit správy, prípadne správy v pull requeste. Táto správa obsahuje (v uvedenom poradí):

- K akému problému / úlohe sa commit viaže (meno, ak nie je, tak stručný a zreteľný popis)
- Čo mal commit riešiť
- Stav, v akom je commitovaná verzia zdrojového kódu
    - Čo bolo vyriešené
    - Čo zostalo nevyriešené
- Commit správy sa píšu v angličtine

Proces: Práca s repozitárom s hrou (eduvirtual-globe):

- Vetva **master** - obsahuje odprezentované a schválené verzie aplikácie
    - Commit je možný iba po konci sprintu, ak product owner potvrdí splnenie definition of done
    - Vo výnimočných prípadoch je možný commit aj inokedy, ak sa zistí chyba v aplikácii, ktorú je treba rýchlo odstrániť
    - Každý commit, ktorý splnil DOD, je označený tagom vX.Y, kde Y sa zvýši po splnení DOD a potvrdení od product ownera, X sa zvyšuje pri vykonaní zásadnej zmeny / pridaní dôležitej funkcionality
- Vetva **devel** - obsahuje rozpracovanú verziu aplikácie, do ktorej je možné vykonávať commity po vyriešení úlohy / po dokončení ucelenej funkcionality
- V prípade práce na väčšej zmene / vývoji zložitejšej funkcionality si zodpovedný vývojár vytvorí vetvu z devel vetvy, označí ju názvom, z ktorého bude jasne vyplývať účel takejto vetvy (preferovaný je underscore-case)
    - Po ukončení práce na funkcionalite sa vyhradená vetva spojí (merge) s devel vetvou
    - Spája ju buď vývojár, alebo teamleader podľa dohody
    - So spojením musia súhlasiť všetci zainteresovaní členovia
    - Po úspešnom spojení sa vyhradená vetva odstráni

Proces: Práca s repozitárom edukačnej stránky (eduvirtual-eduweb):

- Vetva **master** - obsahuje dokončené verzie stránky
    - Commit je možný vykonať v prípade potreby po pridaní ucelenej funkcionality
- Počas vývoja zložitejšej časti stránky sa vytvorí vyhradená vetva z vetvy master

    - Po ukončení práce na funkcionalite sa vyhradená vetva spojí (merge) s master vetvou
    - Spája ju buď vývojár, alebo teamleader podľa dohody
    - So spojením musia súhlasiť všetci zainteresovaní členovia
    - Po úspešnom spojení sa vyhradená vetva odstráni

Proces: Práca s repozitárom webovej stránky projektu (eduvirtual-web):

- Vetva **master** - obsahuje dokončené verzie stránky
    - Commit je možný vykonať v prípade potreby po pridaní ucelenej funkcionality
- Počas vývoja zložitejšej časti stránky sa vytvorí vyhradená vetva z vetvy master
    - Po ukončení práce na funkcionalite sa vyhradená vetva spojí (merge) s master vetvou
    - Spája ju buď vývojár, alebo teamleader podľa dohody
    - So spojením musia súhlasiť všetci zainteresovaní členovia
    - Po úspešnom spojení sa vyhradená vetva odstráni

Proces: Tvorba dočasných repozitárov

- V prípade potreby je možné operatívne vytvárať ďalšie repozitáre
- Pomenovanie repozitárov: eduvirtual-*TYPE_SPECIFICATION*
    - TYPE značí obsah, ktorý sa v repozitáry nachádza (globe pre VR hru, web pre stránku projektu, eduweb pre edukačnú stránku a pod.)
    - SPECIFICATION spresňuje, k čomu repozitár slúži (legacy pre uschovanie zdedeného kódu, revXY pre uloženie revízie a pod.)
- Spôsob tvorby a označovania commitov je zhodný s ostatnými repozitármi, práca s vetvami sa určí dohodou a umiestni sa do wiki stránky repozitára
- Každý člen, ktorému je pridelené prístupové právo je oboznámený s účelom repozitára a spôsobom práce s repozitárom

- Je možné vytvoriť aj read-only repozitáre (nie je umožnený commit ani tvorba novej vetvy)
    - To je užitočné napríklad pri zdieľaní staršej verzie zdedeného kódu - tá má mať iba informatívny charakter

Metodika vývoja C# kódu

verzia 2017-11-02

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá písania programového kódu v jazyku C#. Tieto pravidlá je nutné dodržiavať vo všetkych častiach projektu.

2. Znenie metodiky

Proces: Písanie C# kódu

- Najdôležitejšie pravidlo: pokiaľ nerobím grafické výpočty ktoré sa spúšťajú každý frame tak čitateľnosť kódu a jeho pochopiteľnosť sú dôležitejšie ako dobrý výkon. Ak robím kód náročný na výkon, môžem ho napísať horšie pochopiteľne ale treba okomentovať čo a ako kód robí!!!
    - Nesprávne:

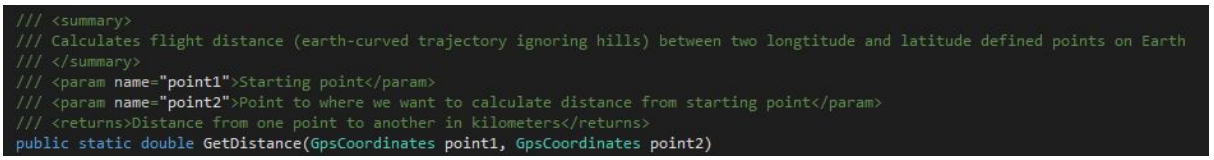
```
public static double GetDistance(GpsCoordinates point1, GpsCoordinates point2)
{
    return GlobalConstants.EARTH_RADIUS_METERS * 2 * Math.Atan2(Math.Sqrt(Math.Sin(ConvertToRadians(point2.Latitude - point1.Latitude) / 2)
        * Math.Sin(ConvertToRadians(point2.Latitude - point1.Latitude) / 2) +
        Math.Cos(ConvertToRadians(point1.Latitude)) * Math.Cos(ConvertToRadians(point2.Latitude)) *
        Math.Sin(ConvertToRadians(point2.Longitude - point1.Longitude) / 2) * Math.Sin(ConvertToRadians(point2.Longitude - point1.Longitude) / 2)),
        Math.Sqrt(1 - Math.Sin(ConvertToRadians(point2.Latitude - point1.Latitude) / 2) * Math.Sin(ConvertToRadians(point2.Latitude - point1.Latitude) / 2) +
        Math.Cos(ConvertToRadians(point1.Latitude)) * Math.Cos(ConvertToRadians(point2.Latitude)) *
        Math.Sin(ConvertToRadians(point2.Longitude - point1.Longitude) / 2) * Math.Sin(ConvertToRadians(point2.Longitude - point1.Longitude) / 2)));
}
```

    - Správne

```
public static double GetDistance(GpsCoordinates point1, GpsCoordinates point2)
{
    float radius = GlobalConstants.EARTH_RADIUS_METERS;

    double latitude1Rad = ConvertToRadians(point1.Latitude);
    double latitude2Rad = ConvertToRadians(point2.Latitude);

    double deltaLatitudeRad = ConvertToRadians(point2.Latitude - point1.Latitude);
    double deltaLongitudeRad = ConvertToRadians(point2.Longitude - point1.Longitude);

    double a = Math.Sin(deltaLatitudeRad / 2) * Math.Sin(deltaLatitudeRad / 2) +
            Math.Cos(latitude1Rad) * Math.Cos(latitude2Rad) *
            Math.Sin(deltaLongitudeRad / 2) * Math.Sin(deltaLongitudeRad / 2);
    double c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

    return radius * c;
}
```

-
-

- Napísať summary funkcie (Visual studio po napísaní /// automaticky doplní snipet s popisom funkcie, existujúcimi argumentmi a návratovou hodnotou. Toto summary sa následne zobrazuje pri mouseoverovaní funkcie/parametrov, vďaka čomu je možné pochopiť čo funkcia vyžaduje a čo spraví bez študovania jej kódu.



```
/// <summary>
/// Calculates flight distance (earth-curved trajectory ignoring hills) between two longtitude and latitude defined points on Earth
/// </summary>
/// <param name="point1">Starting point</param>
/// <param name="point2">Point to where we want to calculate distance from starting point</param>
/// <returns>Distance from one point to another in kilometers</returns>
public static double GetDistance(GpsCoordinates point1, GpsCoordinates point2)
```

Proces: Pomenovávanie premenných, práca s premennými

- POUŽÍVAŤ { get; set; } namiesto java getter setter. get a set môžu obsahovať aj implementáciu ale väčšinou ju netreba. Ak ju treba (príklad):
    - Verejný getter, privátny setter, getter obsahuje úpravu údaju
    - double debt { public get { return Math.Round(field, 2); }; private set; }
    - Ak je logika príliž rozsiahla, je možné ju napísať ako funkciu na viac riadkov.
- Názvy:
    - Triedy: čo najkratšie názvy (public class UpperCamelCase{})
    - Privátne premenné, lokálne premené: začínajú s malým písmenom, ostatné začiatky slov s veľkým (private int lowerCamelCase;)
    - Funkcie: UpperCamelCase
    - Interface: I<UpperCamelCase>
- Neinicializovať fieldy v classach, sú automaticky nastavené na 0/null/false. Pokiaľ potrebujeme inú hodnotu, ktorá nezávisí od argumentov konštruktorov, vtedy môžeme použiť inicializáciu pri deklarácii.
- Operátory: oddeľovať medzerou (1 + x) * 3 || y << 2 namiesto (1+2)*3||y<<2
- Blokové zátvorky na samostatný riadok

- Pokiaľ mám príliš dlhý riadok, rozdeliť ho na viac s odsadením jedného tabu:

```
double a = Math.Sin(deltaLatitudeRad / 2) * Math.Sin(deltaLatitudeRad / 2) +
        Math.Cos(latitude1Rad) * Math.Cos(latitude2Rad) *
        Math.Sin(deltaLongitudeRad / 2) * Math.Sin(deltaLongitudeRad / 2);
```

- Pri konvertovaní primitívnych typov (int->double, float->string) použiť namiesto double x = (double)y; class Convert -> var x = Convert.ToDouble(y). Convert je lahšie debugovateľný pri prípadnej strate dát, ľahšie sa chápe zámer kódu.
- Používať var namiesto explicitného názvu premennej pokiaľ pravá strana výrazu jednoznačne určuje typ výsledku:
    - foreach (var leaf in Tree.Leafs) namiesto foreach (Leaf leaf in Tree.Leafs)
- Garbage collector odstráni objekt len ak na neho neexistuje referencia. Ak pridáte objekt do listu, nezabudnite ho odstrániť.

Proces: Funkcionálna dekompozícia

- C# má JIT compiler (Just in Time), čo znamená, že kompiluje kód za behu programu. Toto umožňuje optimalizáciu strojového kódu podľa hardvérovej špecifikácie systému, na ktorom je program vytvorený. Ak vytvárame veľké bloky kódu, strácame výhody JIT compileru a spôsobujeme pomalší beh programu (musí sa dlho čakať na kompiláciu). JIT kompiluje celý kód funkcie. Funkcie by mali byť čo najkratšie. Ak je vo funkcií veľa if logiky, je lepšie ju nahradiť funkciami alebo objektami. Ak sa funkcionalita opakuje, treba ju vytiahnuť do samostatnej funkcie, pretože táto funkcia po skompilovaní už bude pripravená pri každom jej volaní, namiesto kompilácie tejto logiky na rôznych miestach ako podčasť inej funkcie.

Metodika vývoja HTML_CSS_JS kódu

verzia 2017-11-12

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá písanie HTML, CSS a JavaScript kódu. Metodika tiež stanovuje spôsob organizácie súborov a pravidlá pre tvorbu a zmenu obsahu tímovej stránky.

2. Znenie metodiky

Proces: Pomenovávanie HTML, CSS a JS súborov

Názvy súborov, priečinkov, tried a identifikátorov uvádzame malým písmom v angličtine a medzery nahrádzame znakom "_" prípadne "-". Snažíme sa však udržiavať rovnakú konvenciu v názvoch súborov.
Príklad: **example_main.css** alebo **example-main.css**

Proces: Tvorba hierarchickej štruktúry súborov, organizácia súborov

- Písma v priečinku **fonts**
- CSS štýly v priečinku **css**
- JavaScripty v priečinku **js**
- Obrázky v priečinku **images**
- Videa v priečinku **videos**
- v prípade ak sa dajú vyššie spomínané súbory ďalej organizovať do podpriečinkov tak sa je vhodné ich takto organizovať, v prípade iných súborov postupujeme analogicky

Proces: Písanie HTML a CSS kódu

- Pri písaní HTML kódu usilujeme o dodržiavanie atribútov pre HTML5 a zachovanie sémantiky kódu - elementy, ktoré špecifikujú svoj obsah ako je <form>, <table>, <article>, , … by mali obsahovať len obsah na, ktorý sú určené
- Kaskádové štýly (CSS) ukladať do externých súborov mimo HTML v separátnom priečinku s názvom **css**, rovnako sa vyhýbať inline štýlom ak je to možné, definovanie pozadia (background-image) a podobne sa môžu vyskytovať

  Príklad **inline** štýlu: *** Text ***
- Názvy štýlov by mali byť organizované do hierarchie ako **.parent .child .subchild** { /* Some stuff… */ } namiesto .**parent-child-subchild** { /* Some stuff… */ }, tento spôsob umožňuje jednoduchšie vykonanie prípadných zmien a šetrí miesto v HTML kóde. Príklad je možné vidieť na nasledujúcom obrázku:

- Podľa možnosti využívame CSS atribúty, ktoré sú kompatibilné s viacerými prehliadačmi a vhodné je taktiež definovanie fallback atribútov - napríklad pre

písma “font: 12px Roboto-Light, Arial” ak nie je možné použiť písmo **Roboto** bude namiesto neho použité písmo **Arial**

- Veľké množstvo atribútov je možné zapísať v skrátenom tvare do jedného riadku, ak je to možné a nespôsobuje to neprehľadnosť využívame skrátený tvar. Príklad:

- Vrámci CSS štýlu oddeľujeme súvislé bloky voľným riadkom pre lepšiu prehľadnosť
- Pri mapovaní štýlu preferujeme triedy
  CSS “**.parent**” -> HTML “<p **class=”parent”**></p>” pred identifikátormi
  CSS “**#parent**” -> HTML “<p **id=”parent”**></p>”
- Ak je to možné, minimalizujeme počet definovaných tried tak, že uvedieme názov tagu namiesto definovania triedy tak ako je možné vidieť na nasledujúcom obrázku:

Proces: Písanie JavaScript kódu

- Názvy súborov, metód, objektov, premenných uvádzame v angličtine.
- Okrem inicializácie objektov / funkcií sa musia všetky kódy nachádzať v externých súboroch v separátnom priečinku s názvom **js** - podporuje to jednoduchšie vykonávanie zmien, kde stačí upraviť jeden JS skript namiesto všetkých HTML súborov, kde sa nachádza daný interný skript - podla možnosti by sa však aj inicializačné skripty mali nachádzať v externom súbore
- Pri programovaní skriptov využívame **objekty** kde sa dá namiesto procedurálneho prístupu
- Pri názvoch premenných a funkcií sa využíva **camelCase**
- Pri definovaní názvov “tried” využívame **PascalCase**
- Názvy globálnych premenných a konštánt uvádzame VEĽKÝM_PÍSMOM a ak sa jedná o viac slov oddelíme ich znakom “_”
- Medzi operátormi nechávame medzeru: a = b; a = b + 3; if (a > b)

Metodika vývoja SQL kódu

verzia 2017-11-09

---

1. Dedikácia

Metodika je určená všetkým členom tímu 4 (EduVirtual). Metodika určuje pravidlá pre tvorbu SQL kódu a pomenovávanie dátových entít.

2. Znenie metodiky

Proces: Písanie SQL kódu

- Entity sa pomenúvajú po anglicky, názvom, ktorý čo najjasnejšie vystihuje ich zmysel alebo funkciu
    - Využíva sa PascalCase - názov sa začína veľkým písmenom, pokračuje malými, nové slovo sa napája bez medzery na to predchádzajúce, ale prvé písmeno nového slova je veľké
    - Príklad: User, Course, CourseLevel, CourseUser
- Atribúty v entitách sa píšu malými písmenami, slová sa oddeľujú podtržníkom
    - Príklad: id, user_id, bounding_box