Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Prehliadka kódov v tímových projektoch

(CodeReview)

Dokumentácia k inžinierskemu dielu

Tím: Lucky Seven

Vedúci projektu: Ing. Karol Rástočný

Kontakt:*tp.1314.07@gmail.com*

Ak. Rok : 2013/2014

Autori: Bc. Zuzana Grešlíková

Bc. Matej Chlebana

Bc. Tomáš Kepič

Bc. Patrik Oriskó

Bc. Patrik Samuhel

Bc. Michael Scholtz

Bc. Július Skrisa

OBSAH

1 ÚVOD

Tento dokument obsahuje technickú dokumentáciu k projektu CodeReview implementovanom tímom č. 7 na predmete tímový projekt 1 na Fakulte informatiky a informačných technológií STU v Bratislave.

1.1 ŠTRUKTÚRA DOKUMENTU

Vývoj je riadený metódou „scrum“, preto sa dokument sa skladá z kapitol, ktoré opisujú jednotlivé šprinty. V každom šprinte sú opísané funkcie („user stories“), ktoré boli v rámci neho implementované. Jednotlivé funkcie sú rozdelené na časti: Analýza, Návrh, Riešenie, Testovanie.

1.2 ZADANIE PROJEKTU

Úlohou projektu je vývoj systému pre podporu študentských softvérových projektov. Tento informačný systém bude vytvárať prostredie, v ktorom si budú môcť študenti navzájom zdieľať svoje repozitáre zdrojových kódov a navzájom ich prehliadať, vyjadrovať sa k nim. Budú tak mať možnosť rozvíjať svoje schopnosti prostredníctvom efektívnej a priamej spätnej väzby ku kvalite svojich zdrojových kódov a problémom, ktoré identifikujú počas práce na projekte. Tento systém taktiež zjednoduší spoluprácu tímov a odovzdávanie a priebežnú kontrolu projektov a zadaní.

Cieľom projektu je zjednodušiť a zrýchliť vývoj softvérového produktu, taktiež skvalitniť zdrojový kód vzájomnou spätnou väzbou a dobrou komunikáciou medzi vývojármi softvérového produktu. Webový portál by mal byť využívaný ako tímový nástroj kde členovia tímu komunikujú a kontrolujú kvalitu vytvoreného kódu.

2 STANOVENIE CIEĽOV

Cieľom nášho tímového projektu je vytvoriť systém na prehliadku zdrojových kódov. Systém bude vytváraný ako webová aplikácia s využitím technológie ASP.NET MVC 4. Pomocou webového rozhrania bude možné spravovať softvérové projekty. V našom systéme máme v pláne využiť metódu značkovania zdrojových kódov z programu PerConIK. Túto metódu budeme implementovať do webového zohrania, kde bude možné značkovanie priamo vo webovom prehliadači.

Systém by mal slúžiť na zjednodušenie vyhľadávania chýb v kóde, komentovanie kódu a zvýšenie kvality kódu. Značkovanie zdrojového kódu má veľký význam pre obsiahle projekty, v ktorých sa často po dlhšej dobe stráca prehľadnosť.

Cieľom práce v zimnom semestri je vytvorenie funkčného prototypu. Prototyp by mal spĺňať základnú myšlienku zadania ale nemusí obsahovať všetky navrhnuté komponenty pre finálnu verziu projektu. Metódou vývoja SCRUM budeme vytvárať nový systém a postupne pridávať funkcionalitu. Každá iterácia bude obsahovať nové požiadavky na systém, ktoré budú jednotlivý členovia tímu postupne riešiť a tým vytvárať zjednotený systém.

V rámci zimného semestra bolo našim cieľom taktiež zlepšenie tímovej práce a zosúladenie prác jednotlivých členov tímu. Na základe vytvorenia niekoľkých metodík špecifikujeme konkrétne postupy pri práci na rôznych častiach projektu či už sa jedná o písanie kódu, verziovanie, vytváranie dokumentácie a i.

Webová aplikácia by mala obsahovať možnosť prideľovania projektov administrátorom. Následne budú používatelia systému môcť pracovať s jednotlivými pridelenými projektmi. V projekte bude možné zobraziť celkovú štruktúru súborov a entít projektu. Systém dokáže zobraziť rôzne verzie projektu a rozdiely medzi týmito verziami. Pri prehliadke kódov bude následne možné kód komentovať a pridávať značky rôzneho typu a obsahu. Ďalšia funkcionalita bude pribúdať v priebehu vývoja

3 PRED PRVÝM ŠPRINTOM

Prvou úlohou tímu bolo rozdelenie zodpovedností medzi členov. Zodpovednosti (úlohy) boli roztriedené členmi podľa predošlých skúseností a špecifikácie daných úloh.

Keďže náš tím sa zaoberá vývojom webového prostredia pre systém AST-RCS, ktorý spravuje projekty, ich verzie, používateľov, bolo nutné sa oboznámiť s týmto systémom. Systém tvorí základ pre správu dát, ktoré môžu slúžiť na zdokonalenie prehliadok kódu ale aj samotného vývoja softvérového produktu.

Pred prvým šprintom bolo potrebné naštudovať dokumentáciu systému, ako aj oboznámiť sa s cieľmi celého projektu. Tieto úkony boli potrebné na rýchle prispôsobenie členov tímu na diskusie a návrhy, ako aj na rýchlejší nábeh na vývojové tempo v projekte. Hlavným cieľom štúdia systému, bolo porozumieť logike načítavania údajov, ako aj spoznať najefektívnejšie spôsoby implementácie.

Pred prvým šprintom bolo taktiež zabezpečované samotné vývojové prostredie, či už individuálne na počítačoch členov tímu. Kde bolo nutné inštalovať niekoľko softvérov podporujúcich vývoj. Taktiež sa zabezpečoval a vyjednával server, ktorý bol nutný pre začiatok vývoja.

4 SEMIENKO

Číslo šprintu: 1

Začiatok šprintu: 9.10.2013

Koniec šprintu: 23.10.2013

Príbehy:

- **Prihlásenie používateľa AIS loginom**
- **Pridanie informácií o tíme na stránku projektu**
- **Určenie administrátora systému**
- **Priradenie používateľa k projektu**
- **Zobrazenie zoznamu projektov používateľa**
- **Načítanie súborovej štruktúry projektu do stromovej reprezentácie**
- **Načítanie AST štruktúry projektu do stromovej reprezentácie**
- **Prehliadanie stromovej reprezentácie**
- **Zobrazenie zdrojového kódu zvolenej entity z AST-RCS**
- **Zmena hesla používateľa**
- **Registrácia emailu používateľa**
- **Zvýraznenie syntaxe zobrazeného kódu**
- **Zobrazenie zoznamu odovzdaní projektu**

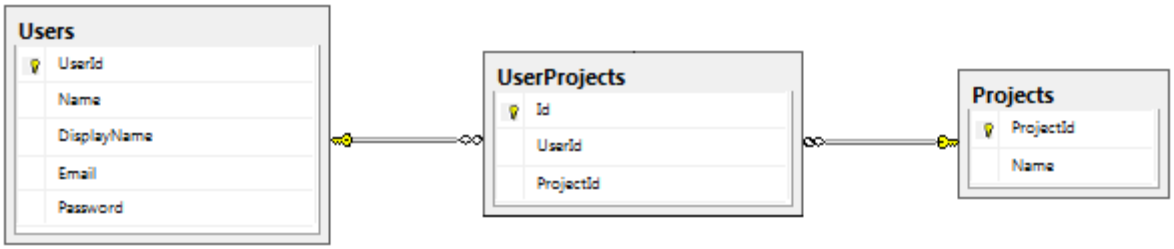
4.1 PRIHLÁSENIE POUŽÍVATEĽA AIS LOGINOM

Používateľ sa môže prihlásiť prostredníctvom *AIS* konta, aby mohli študenti STU používať vyvíjaný systém. Registrácia do systému *CodeReview* prebehne po úspešnom overení používateľa voči systému *AIS*.

Riešenie

Riešenie tohto príbehu malo viacero nadväzujúcich aktivít. Keďže sa jednalo o jednu z prvých úloh, bolo v prvom rade potrebné nainštalovať a nakonfigurovať server, nainštalovať databázový server a vytvoriť databázu. Pre jednoduchší prístup k dátam v databáze sme použili objektovo relačný mapovač *LinqToSQL*.

Dátový model obsahuje tabuľku *Users* a tabuľky pre uloženie projektov.



Obr. č. 1. Databázový model

Users – Tabuľka obsahuje základné údaje o registrovaných používateľoch

- *UserId* – identifikačné číslo používateľa
- *Name* – prihlasovacie meno používateľa (prihlasovacie meno do AIS)
- *DisplayName* – celé meno používateľa, zobrazované na stránke (meno študenta z AIS)
- *Email* – email používateľa (meniteľné, štandardne získané z AIS)
- *Password* – kryptované heslo používateľa

Projects – Tabuľka projektov, čerpaných z AST-RCS

- *ProjectId* – identifikačné číslo projektu čerpané zo systému AST-RCS
- *Name* – názov projektu čerpané zo systému AST-RCS

UserProjects – entitno relačná tabuľka, ktorá obsahuje záznamy o pridelených projektoch používateľom

Na overenie používateľa voči systému *AIS*, používame *OpenLDAP* server. Konfigurácia *AIS OpenLDAP* je nasledovná:

```
<setting name="LDAPEndpoint" serializeAs="String">
  <value>LDAP://ldap.stuba.sk:389/ou=People,dc=stuba,dc=sk</value>
</setting>
<setting name="LDAPUserDN" serializeAs="String">
  <value>uid={0},ou=People,dc=stuba,dc=sk</value>
</setting>
```

Obr. 2. Konfigurácia *OpenLDAP*

Prihlasovanie prebieha prostredníctvom prihlasovacieho formulára cez *http forms authentication*. Používateľ zadáva prihlasovacie meno a heslo. Heslo musí mať minimálnu dĺžku 6 znakov. Spojenie medzi našou aplikáciou a *OpenLDAP* serverom prebieha len v prípade ak databáza ešte neobsahuje meno používateľa, ktorý sa snaží prihlásiť. Po úspešnom overení používateľa cez *AIS OpenLDAP,* sa

vytvorí nový používateľ v internej databáze a viac sa voči *AIS* neoveruje. Heslá sú ukladané v databáze a sú kryptované 128 bitovou hešovacou funkciou *MD5*.

4.2 Pridanie informácií o tíme na stránku projektu

Na stránke systému *CodeReview* bolo potrebné zobraziť informácie o projekte a o našom „*Lucky seven*“ tíme.

Riešenie

Pridali sme novú stránku s názvom *About*. Na stránku sme umiestnili mená vývojárskeho tímu ako aj odkaz na webovú stránku tímu.

Obr. 3. Informácie o tíme

4.3 Určenie administrátora systému

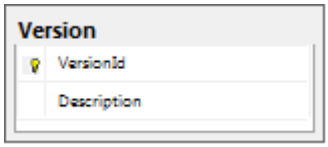
V databáze systému *CodeReview* je možné priradiť používateľovi rolu administrátora, vďaka čomu bude môcť administrátor vykonávať administračné úkony.

Analýza

Museli sme pridať ďalšiu tabuľku na definovanie používateľských rolí. Očakávame, že používatelia budú môcť mať súčasne viacero rolí. Z tohto dôvodu sme museli vytvoriť entitno relačnú tabuľku, ktorá umožní pridelenie ľubovoľného počtu rolí používateľom. Pre zmenu databázového modelu bolo potrebné spraviť zásah do databázovej štruktúry. Aby sme zakaždým nemuseli generovať novú databázu, vytvorili sme migrácie, ktoré umožňujú kontinuálne zmeny v databáze. Migrácie sa aplikujú pre každú zmenu vykonanú nad dátovým modelom.

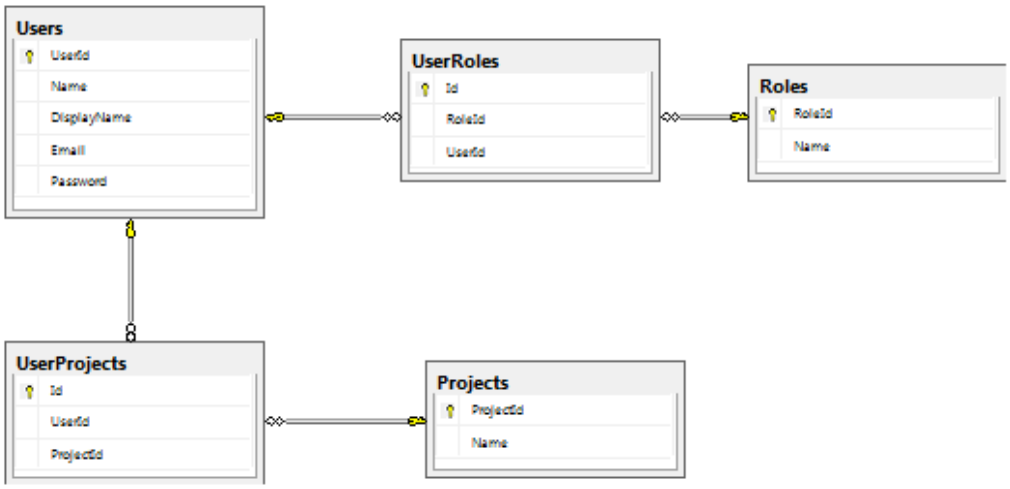
Riešenie

Keďže sme chceli meniť databázu bez silného zásahu do systému, museli sme aplikovať migrácie. Pri spustení stránky, systém načíta migračné scripty a spustí tie, ktoré ešte neboli aplikované. Aby systém vedel či má, alebo nemá aplikovať migráciu, musí si zaznamenávať verziu databázy. Pre zaznamenávanie verzie databázy sme museli vytvoriť novú tabuľku s názvom *Version*. Tabuľka obsahuje *Id* verzie a stručný opis zmien, ktoré boli vykonané v aktuálnej verzií.



Obr. č. 4. Tabuľka verzií

Dátový model v tomto štádiu obsahoval iba tabuľku používateľov s ich projektmi, preto sme pridali do súčasného modelu aj tabuľku používateľských rolí. Pre zmenu modelu sme vytvorili migračný script, ktorým sa vytvorili nové tabuľky a cudzí kľúč na vytvorenie prepojenia s tabuľkou *Users*.



Obr. č.5 Dátový model rozšírený o roly používateľov

Roles – Tabuľka číselníkov, obsahujúca roly používateľov (administrátor, používateľ)

- *RoleId* – identifikačné číslo roly
- *Name* – názov roly

UserRoles – entitno relačná tabuľka, ktorá obsahuje záznamy o roliach používateľov

4.4 PRIRADENIE POUŽÍVATEĽA K PROJEKTU

Nie všetci používatelia majú prístup ku všetkým projektom. Každý používateľ má prístup len ku projektom, ktoré mu administrátor priradí. Preto je nevyhnutné, aby mohol administrátor k projektu priradiť používateľa, čím by používateľ získal prístup k danému projektu a mohol by s ním pracovať.

Návrh

V prvom kroku je potrebné načítať zoznam všetkých projektov a zoznam všetkých používateľov. Tieto dáta je potrebné uložiť do modelu a daný model zobraziť používateľovi v prehľadnom „view“.

Po načítaní údajov sa administrátorovi zobrazí list používateľov a list projektov, v ktorom označí používateľa a projekt, ktorý mu chce priradiť. Po zvolení používateľa a projektu administrátor môže daný výber uložiť. Po uložení sa tieto údaje uložia do databázy.

Riešenie

Projekty sú načítavané z AST-RCS pomocou metódy

```
private List<Project> ReadProjectsFromASTRCS()
            {
                  var projectlist = new List<Project>();

                  using (var client = ServiceClient.AstRcsClient)
                  {
                        var projectsets = client.SearchRcsProjects(new
SearchRcsProjectsRequest());

                        foreach (var rcsproject in projectsets.RcsProjects)
                        {
                              projectlist.Add(new Project()
                              {
                                    Name =
rcsproject.Url.Substring((rcsproject.Url.LastIndexOf("/") + 1)),
                                    ProjectId = rcsproject.Id
                              });
                        }
                  }
                  //Vlozenie projektov z AST-RCS do databazy
                  ProjectMethods.InsertAllProjects(projectlist);

                  return projectlist;
            }
```

V tejto metóde prebehne načítanie projektov a porovnanie s aktuálnou databázou. V prípade, že sú v AST-RCS nové projekty, ktoré neboli zatiaľ do databázy pridané, sú vložené do databázy.

V ďalšom kroku sú z databázy načítaní všetci používatelia s rolou „user“ .

Tieto údaje sa administrátorovi zobrazia v dvoch „listbox-och“. Po uložení ním zadaných údajov je používateľovi s rolou „user“ priradený vybraný projekt a tieto údaje sú uložené do databázy.

4.5 ZOBRAZENIE ZOZNAMU PROJEKTOV POUŽÍVATEĽA

Cieľom bolo umožniť používateľovi prístup ku svojím projektom. Na to, aby mohol používateľ pristúpiť k danému projektu je nevyhnutné mu zobraziť všetky jeho projekty, z ktorých si následne má možnosť zvoliť ten projekt, na ktorom chce aktuálne pracovať.

Návrh

Projekty prihláseného používateľa je potrebné načítať z databázy . Vytvoriť model pre uloženie týchto projektov a zobraziť ich prehľadne používateľovi.

Riešenie

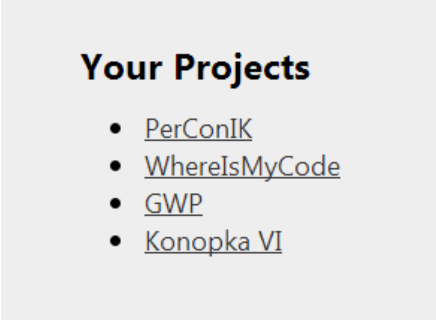
Na načítanie projektov z databázy je použitá metóda

```
public static List<Project> GetUserProjects(User user)
        {
            using (var dc = new CRDataContext())
            {
                return dc.UserProjects.Where(x => x.UserId ==
user.UserId).ToList().Select(x => x.Project).ToList<Project>();
            }
        }
```

Metóda vráti list projektov používateľa „user“. List projektov je vložený do modelu

```
public class UserProjects
  {
    public IList<Project> UserProjectsList { get; set; }
  }
```

A následne sú vo „view“ používateľovi jeho projekty zobrazené, ako na obrázku č. 6.



Obr. č. 6 Zobrazenie projektov používateľa

Testovanie

Testovanie nebolo v tejto fáze riešené pomocou automaticky naprogramovaných testov. Výsledky boli testované manuálne porovnávaním s údajmi v databáze.

4.6 NAČÍTANIE SÚBOROVEJ ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE

Pre zobrazenie súborov projektu sme sa rozhodli zobrazovať štruktúru súborov v strome. Takáto štruktúra je prehľadná s ľahkým vyhľadávaním.

Analýza

Je potrebné analyzovať prístupy k tvorbe stromovej štruktúry v ASP.NET MVC 4. Analyzujeme spôsoby vytvárania stromu v jazyku C#. Následne analyzujeme metódy zobrazenia stromovej štruktúry v cshtml a css.

Návrh

Podľa analýzy sme navrhli jednoduchú stromovú štruktúru pozostávajúcu s uzlov, ktoré obsahujú informácie o jednotlivých uzloch a zoznam referencii na ďalšiu úroveň uzlov.

Riešenie

Pre reprezentáciu uzla bola vytvorená trieda EntityNode.

Zo systému AST-RCS boli pomocou servisu SearchFiles zaznamenané cesty ku všetkým súborom požadovaného projektu. Z ciest k súborom bola vytvorená stromová štruktúra súborov projektu.

Objekt stromu bol následné zobrazený ako zoznam na stránke. Strom sa menil podľa zadaného čísla projektu.

Testovanie

Zobrazená stromová štruktúra bola porovnávaná s reálnou štruktúrou súborov v niekoľkých projektoch. Zobrazené štruktúry sa zhodovali.

4.7 NAČÍTANIE AST ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE

V tomto príbehu je nutné načítať kódové entity zo systému AST-RCS, tak aby ich bolo možné zobraziť používateľovi v stromovej štruktúre. Je takisto potrebné načítať podrobné údaje o týchto kódových entitách ako názov, typ a iné údaje, ktoré by mohli používateľa zaujímať.

Analýza

V prvom rade je nutné analyzovanie možností systému AST-RCS, z ktorého je nutné načítať údaje. Z tohto systému je potrebné nahrať údaje o všetkých kódových entitách daného projektu. Vstupom do tejto funkcie je projekt, ktorého entity chceme načítať. K dispozícií sú servisy, ktoré poskytujú údaje z databázy AST-RCS a mnoho vyhľadávacích metód, ktoré je možné použiť.

Metódy servisu súvisiace s kódovými entitami:

GetCodeEntity() – vráti najnovšiu entitu a informácie o nej podľa jej ID.
GetCodeEntityHistory() – vráti predchodcov entity podľa jej ID a intervalu.
SerachCodeEntity() – hľadá entity podľa rôznych kritérií a vráti pole ID vyhovujúcich entít.

Návrh

Pri riešení tohto problému je nutné efektívne využiť metódy poskytované systémom AST-RCS a údaje zoradiť do stromovej štruktúry.

Pre načítanie všetkých entít daného projektu je možné využiť funkciu *SerachCodeEntity(),* kde je možné podľa identifikátora projektu vyhľadať všetky projektové kódové entity. Metóda vráti zoznam identifikátorov entít.

Načítané údaje je neskôr možné využiť v ďalšej funkcií (*GetCodeEntity()*), ktorá načíta podrobné údaje o každej entite podľa identifikátora. Podľa týchto podrobných údajov bude možné zostrojiť stromovú štruktúru.

Riešenie

V riešení sú využité navrhnuté metódy zo systému AST-RCS. Po zavolaní týchto metód sú načítané všetky kódové entity s veľkým množstvom atribútov. Na zostrojenie stromovej štruktúry sú potrebné nasledovné atribúty:

- *ID – identifikátor entity v rámci systému AST RCS.*
- *ParentEntityId – identifikátor rodiča kódovej entity.*
- *EntityType – identifikátor typu entity.*

Po načítaní podrobných údajov o každej entite do triedy *EntityNode* sú údaje spracované do dátovej štruktúry *Dictionary<int, EntityNode>,* tak aby podľa identifikátorov bolo možné pristupovať k jednotlivým entitám.

Iteráciou cez túto štruktúru program vyplní údaje o potomkoch entít v štruktúre EntityNode. Týmto krokom sa vytvorí stromová štruktúra. Metóda, ktorá načítava tieto údaje vráti koreň („root“) tohto stromu na ďalšie spracovanie.

Atribúty EntityNode:

```
        public int ID { get; set; }
        public int VersionID { get; set; }
        public int? ParentID { get; set; }
        public string Type { get; set; }
        public string Name { get; set; }
        public List<EntityNode> Children;
```

Testovanie
Testovanie tohto príbehu je testované spolu s príbehom, ktorý zobrazuje tieto údaje na stránke. Je vytvorený automatický jednotkový test, ktorý sa spúšťa spolu s nasadzovaním kódu.

4.8 PREHLIADANIE STROMOVEJ REPREZENTÁCIE

V tomto príbehu bolo potrebné zabezpečiť vizuálne zobrazenie stromovej štruktúry získanej z verziovacieho systému AST-RCS a reprezentovať ju vizuálne a prehľadne vo webovom prehliadači používateľa. Konkrétne ide o stromové štruktúry, ktoré reprezentujú súborovú a entitnú štruktúru projektu samotného.

Analýza
Ako prvé bolo potrebné zvážiť, akým spôsobom by mohol byť daný strom vizualizovaný vo webovom prehliadači. Keďže ide o súborovú resp. entitnú štruktúru, ako prvé nás napadlo využitie HTML značky neusporiadaný zoznam ("<ul>"). Táto značka má možnosť v sebe obsahovať vnorené neusporiadané zoznamy a takýmto spôsobom reprezentovať stromovú štruktúru, pričom samotné úrovne sú potom jednotne odsadzované. Ďalšou možnosťou by bolo vytvorenie vlastného systému, ktorý by sa musel špeciálne naštýlovať. Vzhľadom na komplexnosť takéhoto systému, ktorý by iba imitoval niečo čo už existuje, bol použitý prvý nápad využívajúci neusporiadaný zoznam.

Návrh
Na zabezpečenie tejto funkcionality bolo potrebné navrhnúť dve samostatné časti.

Sú to:

1. Kontrolér - ovláda logiku; načítanie stromu podľa zvoleného typu
2. View - viaže sa na kontrolér; vizualizuje výsledky

Samotný kontrolér bude využívať kód, ktorý napísali dvaja iný členovia tímu. Tento kód sa stará o načítanie údajov z AST-RCS a vytvorenie dátovej štruktúry. Následne vráti odkaz na koreň stromu, ktorý reprezentuje získané dáta vo vopred definovanej podobe.

Riešenie
Ako prvý bol vytvorený kontrolér s názvom *TreeViewController*. Tento kontrolér prijíma niekoľko parametrov a na základe nich načítava korektný typ údajov o zvolenom projekte. Sú to parametre:

1. type (Integer) - typ stromu; súbory alebo entity
2. projectId (Integer) - identifikačné číslo zvoleného projektu

Typy stromov sú definované v súbore *TreeTypesEnum*. Momentálne existujú dva typy:

- *TreeTypesEnum.CodeEntities* - kódové entity
- *TreeTypesEnum.ProjectFileEntities* - súborové entity

Na základe zvoleného typu je zavolaná príslušná metóda zo statickej triedy *LoadEntities*:

- *LoadEntities.LoadCodeEntities()* - pre kódové entity
- *LoadEntities.LoadFileSystemEntities()* - pre súborové entity

Po prijatí stromovej štruktúry je zavolané View, do ktorého sa celá štruktúra pošle.

Vo View je stromová štruktúra vizualizovaná pomocou vnorených neusporiadaných zoznamov. Keďže je to pomerne komplexná dátová štruktúra, vo View je ešte definovaná pomocná funkcia. Táto funkcia rekurzívne vykreslí strom na základe danej dátovej štruktúry.

Testovanie
Za predpokladu, že kód na načítanie dátovej štruktúry poskytnutý mojimi tímovými kolegami je korektný, bolo potrebné overiť, či prebieha korektne aj samotné vykresľovanie v prehliadači. Overovanie prebiehalo hlavne manuálne, kedy sa porovnávala získaná vizuálna reprezentácia s tou, ktorá sa nachádzala vo verziovacom systéme. Bolo potrebné overiť, či jednotlivé úrovne boli korektne odsadené. Pri overovaní a testovaní finálnej verzie neboli nájdené problémy.

4.9 Zobrazenie zdrojového kódu zvolenej entity z AST-RCS

V tomto príbehu bolo potrebné zabezpečiť zobrazenie zdrojového kódu zvolenej entity z verziovacieho systému AST-RCS pre prehliadanie obsahu danej entity používateľom.

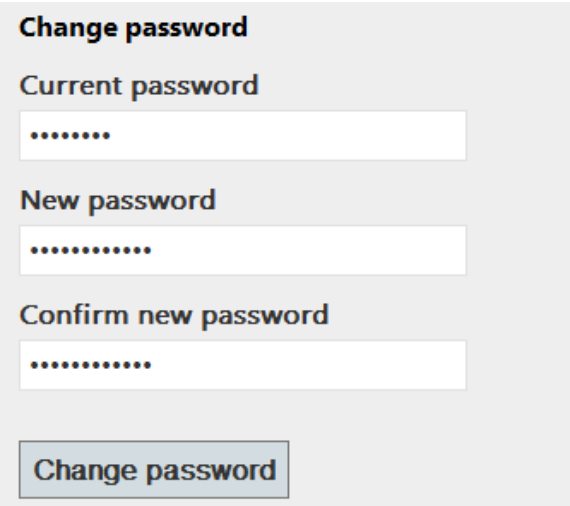
Riešenie

Na zabezpečenie požadovanej funkcionality bola implementovaný kontrolér *FileRequestController*, ktorý slúži na načítanie obsahu zvolenej entity. Tento kontrolér prijíma presný identifikátor verzie (*versionId*) zvolenej entity, na základe ktorej spraví dopyt pre získavanie obsahu entity zo systému AST-RCS. Po načítaní obsahu entity sa zavolá prislúchajúci View, kde je reprezentovaný zdrojový kód entity.

4.10 Zmena hesla používateľa

Používateľ si môže zmeniť svoje heslo do systému, aby zvýšil bezpečnosť svojho konta.

Riešenie
Pre zmenu hesla bolo potrebné vytvoriť formulár na zmenu hesla do systému. Formulár sa zobrazí po kliknutí na meno aktuálne prihláseného používateľa.



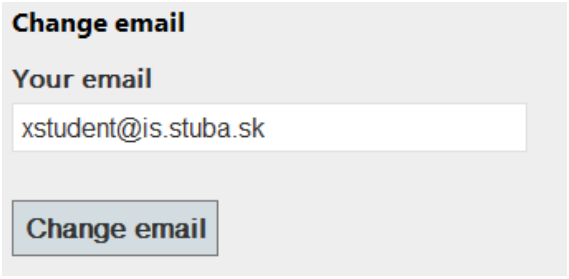
Obr. č. 7 Formulár na zmenu hesla používateľa

4.11 Registrácia emailu používateľa

Používateľ si zaregistruje email, aby mohol v budúcnosti prijímať notifikácie.

Riešenie

Registrácia emailu používateľa prebieha pri registrácií. Email používateľa získame spolu s ostatnými dátami zo servera *AIS OpenLDAP*. Tento email uložíme do databázy k používateľovi. Ďalšou úlohou bolo vytvoriť formulár na zmenu emailu používateľa. Formulár sa zobrazí po kliknutí na meno aktuálne prihláseného používateľa.



Obr. č. 8 Formulár na zmenu emailu používateľa

4.12 Zvýraznenie syntaxe zobrazeného kódu

Úlohou pri tomto príbehu bolo zlepšiť prehľadnosť a uľahčiť čitateľnosť zobrazeného zdrojového kódu zvolenej entity. Na zabezpečenie spomenutého bolo potrebné vytvoriť riešenie na zvýraznenie zdrojového kódu entity.

Analýza

Skúmané boli možné riešenia zvýraznenia zdrojových kódov pre webové stránky. Tieto riešenia sú zvyčajne implementované v podobe JavaScript a CSS súborov, ktoré sa začlenia do HTML kódu danej stránky. Úlohou bolo nájsť najprijateľnejšie riešenie pre náš projekt.

Návrh

Po analýze problematiky bol vybratý JavaScript modul *google-code-prettify*, ktorý umožňuje definovanie vlastných tém pre zvýraznenie zdrojových kódov, číslovanie riadkov zobrazeného kódu, a podporuje zvýraznenie zdrojových kódov pre viaceré programovacie jazyky. Na obrázku č. 9 je

```
<script                                                    src="https://google-code-
prettify.googlecode.com/svn/loader/run_prettify.js">
</script>
<pre class="prettyprint linenums:1">@codeContent</pre>
```

znázornená jeho použitie v súbore .cshtml.

Obr. č. 9 použitie modulu *google-code-prettify* v súbore .cshtml

Riešenie

Implementácia príbehu bola spravená podľa návrhu. Aplikovanie zvýraznenia zdrojových kódov zvolenej entity pri prehliadnutí jej obsahu bolo zahrnuté do „View“ kontroléra *FileRequest*. Podobným spôsobom sa da aplikovať zvýraznenie aj pre ďalšie zobrazenia zdrojového kódu, ktoré môžu pribudnúť v nasledujúcich šprintoch.

Testovanie

Testovanie príbehu prebiehalo manuálne pre rôzne obsahy zvolených entít. V každom prípade sa aplikovalo zvýraznenie obsahu danej entity, test skončilo úspešne.

4.13 ZOBRAZENIE ZOZNAMU ODOVZDANÍ PROJEKTU

Hlavnou úlohou tohto príbehu bolo zobrazenie odovzdaní projektu, tzv. changesetov. Tie slúžia na to, aby si mohol používateľ pozrieť, kedy nastali zmeny v programe – kto a kedy uložil aktuálnu verziu programu na server.

Analýza

Jednotlivé odovzdania programov sú uložené na serveri AST-RCS. Pomocou volania rôznych služieb môžeme zo servera získať zoznam changesetov. Aby sme mohli volať služby, potrebujeme mať najprv vytvoreného klienta, pomocou ktorého budeme získavať údaje. Existuje viacero typov klientov. My použijeme klienta pre pripojenie k AstRcs službám a User službám, pričom medzi AstRcs službami sú volania na získanie changesetov a pri User službách volania pre získanie mena (loginu) používateľa pomocou jeho ID na serveri.

Návrh

V prvom kroku vytvoríme dvoch klientov na pripojenie k serveru, aby sme mohli využívať jeho služby. V druhom kroku získame zoznam konkrétnych changesetov. Tie získame pomocou volania služby v AstRcsService. Pri volaní služby máme možnosť určiť za koľko posledných dní chceme dostať zoznam, tj. maximálne aké staré majú byť changesety. Funkcia vracia iba zoznam ID (identifikačných čísiel changesetov), preto zoznam prejdeme a necháme si vytiahnuť konkrétny changeset s podrobnými informáciami. Informácie následne uložíme do modelu obsahujúceho list changesetov a ten vypíšeme zatiaľ v jednoduchej internetovej stránke.

Riešenie

Najprv sme vytvorili *ChangesetController.cs* s príslušným View.

Pri implementácií sme použili dvoch klientov: *RcsServiceClient* a *UserServiceClient.* Po napojení s RCS klientom sme pomocou služby *SearchChangesets* vyhľadali zoznam ID changesetov, ktoré sa zmenili počas nami posledných zadaných dní. Zdrojový kód volania môžeme vidieť nižšie.

```
// vyhladanie changesetov
var changesets = rcsClient.SearchChangesets(new SearchChangesetsRequest()
{
        TimestampFrom = DateTime.Now.AddDays(-21),
        TimestampFromSpecified = true
});
```

Zoznam ID hodnôt sme prešli v cykle a vyhľadali sme informácie o konkrétnom changesete. Na pomoc pre ukladanie changesetov sme vytvorili model, ktorý obsahuje list skladajúci sa z changesetov a userov, ktorý ho vykonali. Usera (používateľa) ukladáme osobitne z dôvodu, že v informáciách o changesete je iba ID používateľa. Informácie o používateľovi s konkrétnym ID sme získali pomocou služby v UserService s názvom GetUser . Jedna položka listu modelu sa tak skladá z dvoch prvkov a to *ChangesetDto* a *UserDto*, ktoré sú modelmi v AST-RCS. Výsledný model po

spracovaní všetkých nájdených changesetov sme poslali do príslušného View a zobrazili ako jednoduchú tabuľku pomocou jazyka HTML.

Testovanie

V tejto fáze sme testovanie ešte neriešili pomocou naprogramovaných testov. Výsledky, ktoré sme získali, sme porovnávali so zoznamom changesetov, ktorý môžeme nájsť na našom verziovacom a tímovom serveri TFS. Pri testovaní sme pozorovali jeden problém, a tým bolo zobrazovanie changesetov rôznych tímov a nemožnosť filtrácie pri vyhľadávaní.

5 KORIENOK

Číslo šprintu: 2

Začiatok šprintu: 23.10.2013

Koniec šprintu: 6.11.2013

Príbehy:

- **Zobrazenie zoznamu projektov používateľa z novej verzie AST-RCS**
- **Rozbaľovateľná stromová štruktúra**
- **Priradenie používateľa k projektu z novej verzie AST-RCS**
- **Zobrazenie stromovej štruktúry vedľa zobrazenia zdrojového kódu**
- **Zobrazenie grafu odovzdaní projektu**
- **Zobrazenie zoznamu zmien vykonaných vo zvolenom odovzdaní**
- **Zobrazenie zdrojového kódu zmenenej entity v odovzdaní so zvýraznenými zmenami**
- **Načítanie histórie súborovej entity z AST-RCS**
- **Načítanie histórie entity zdrojového kódu z AST-RCS**
- **Zobrazenie histórie entity vo forme tabuľky**
- **Porovnanie zdrojového kódu dvoch zvolených verzií entity z AST-RCS**
- **Načítanie súborovej štruktúry projektu do stromovej reprezentácie pre novú verziu AST-RCS**
- **Načítanie AST štruktúry projektu do stromovej reprezentácie pre novú verziu AST-RCS**
- **Zobrazenie zdrojového kódu zvolenej entity z novej verzie AST-RCS**
- **Zobrazenie zoznamu odovzdaní projektu v novej verzii AST-RCS**
- **Úspešné prihlásenie s neplatným heslom**

5.1 ZOBRAZENIE ZOZNAMU PROJEKTOV POUŽÍVATEĽA Z NOVEJ VERZIE AST-RCS

Pri prechode na novú verziu AST-RCS boli zmenené prístupy k načítaniu údajov z databázy. Podľa novej dokumentácie bolo potrebné upraviť metódy načítania údajov.

Riešenie

V novej verzií AST-RCS bolo pre načítanie potrebné zmeniť klienta. V starej verzii prebiehalo načítavanie projektov nasledovne

```
    var projectsets = c.SearchRcsProjects(new SearchRcsProjectsRequest());

    foreach (var projectid in projectsets.ProjectIds)
    {
        var projectinfo = c.GetRcsProject(new GetRcsProjectRequest()
        {
            RcsProjectId = projectid
        });
    }
    c.Close();
}
```

Pri prechode na novú verziu bolo potrebné kód upraviť na

```
using (var client = ServiceClient.AstRcsClient)
{
    var projectsets = client.SearchRcsProjects(new SearchRcsProjectsRequest());
}
```

5.2 ROZBAĽOVATEĽNÁ STROMOVÁ ŠTRUKTÚRA

V tomto príbehu bolo potrebné zabezpečiť modifikácia zobrazovanej stromovej štruktúry a to takým spôsobom, že všetky uzly stromu sa dajú kliknutím jednoducho zbaliť a rozbaliť. Pri veľkých projektoch to zabezpečí rýchlejšiu a prehľadnejšiu prácu zo stromovou vizualizáciou.

Analýza

Je potrebné zvážiť, akým spôsobom je možné docieliť interakciu v okne prehliadača pri ktorej sa priamo mení obsah zobrazovaného dokumentu. Najrozumnejšie riešenie spočíva v použití JavaScriptu, ktorý po kliknutí na uzol stromu zabezpečí, aby sa jeho potomok skryl prípadne odokryl. Je vhodné zvážiť, či nebude dobré použiť nejakú dostupnú JavaScriptovú knižnicu, ktorá by nám prácu uľahčila, napr. jQuery.

Návrh

Návrh riešenia spočíval v napísaní jednoduchého JavaScriptového kódu, ktorý po kliknutí na daný uzol vykoná akciu, pri ktorej skryje, resp. odokryje svojich potomkov. Pri riešení bola použítá aj JavaScriptová knižnica jQuery, ktorá je priamo podporovaná v nami použitom frameworku MVC.

Riešenie

Pri riešení bol najprv mierne upravený spôsob vykresľovania stromu. Táto úprava nemala vplyv na vizuálnu stránku vykresleného stromu, ale mierne zjednodušila samotné zbaľovanie a rozbaľovanie

stromovej štruktúry. Okrem toho bola všetkým uzlom (okrem listov) pridaná v HTML kóde trieda "toggle". Na základe tejto triedy vie JavaScriptový kód interagovať s uzlami stromu. Samotný JavaScriptový kód je vďaka použitie jQuery knižnice triviálny (obr. č. 10).

```
$(".toggle").click(function () {
        $(this).next().slideToggle();
});
```

Obr. č. 10 Ukážka kódu zabezpečujúceho interakciu so stromom

Testovanie

Testovanie prebiehalo jednoducho základným overením, či sa daný strom rozbaľuje a zbaľuje korektne. Testovaný strom bol porovnávaný s plne rozbalenou verziou stromu.

5.3 PRIRADENIE POUŽÍVATEĽA K PROJEKTU Z NOVEJ VERZIE AST-RCS

Pri prechode na novú verziu AST-RCS boli zmenené prístupy k načítaniu údajov z databázy. Podľa novej dokumentácie bolo potrebné upraviť metódy načítania údajov.

Riešenie

V novej verzii AST-RCS bolo potrebné upraviť načítanie projektov, ktoré je bližšie opísané v bode 5. 1.

5.4 ZOBRAZENIE STROMOVEJ ŠTRUKTÚRY VEDĽA ZOBRAZENIA ZDROJOVÉHO KÓDU

Cieľom tohto príbehu je zabezpečiť, že ak si používateľ klikne v súborovom strome na názov súboru, obsah tohto súboru sa mu zobrazí v druhej časti okna.

Analýza

Je potrebné zistiť, akým spôsobom zabezpečíme, aby sme vedľa seba mali dve separované časti, strom a obsah súboru v rámci jednej obrazovky. Ďalej je potrebné zabezpečiť, aby keď používateľ klikne na názov súboru, obsah tohto súboru sa mu zobrazí na obrazovke bez toho aby nastalo obnovenie stránky.

Návrh

Je potrebné zabezpečiť skombinovanie dvoch samostatných kódov, ktoré už boli vytvorené. Zobrazovanie stromov som riešil už v predchádzajúcom šprinte. Zobrazovanie súborov a zvýraznenie syntaxe riešil v minulom šprinte kolega s tímu. Návrh teda pozostáva s kombinácie týchto dvoch riešení. Pre získavanie obsahu zvoleného súboru bude potrebné použiť AJAXové asynchrónne volania, vďaka ktorým vieme získavať nový obsah a ním aktualizovať dokument bez jeho obnovenia v prehliadači.

Riešenie

Samotné riešenie pozostávalo z opätovného využitia čiastkových riešení, ktoré boli vytvorené v minulom šprinte. Vedľa stromovej štruktúry je vytvorená nová sekcia použitím HTML značky * .* Táto sekcia je pri prvom načítaní prázdna. Po kliknutí na ľubovoľný názov v strome sa načíta obsah zvoleného súboru a spomínaná sekcia sa vyplní. Riešenie obnovuje obsah stránky bez jej reálneho opätovného načítania. O túto funkcionalitu sa stará pomocná AJAX metóda, ktorá je priamo súčasťou MVC. Pri použití takejto metódy nie je potrebné písať vlastný JavaScriptový kód (obr. č. 11).

```
@Ajax.ActionLink(n.Name, "FileRequest",
new RouteValueDictionary {{ "versionId", n.VersionID }, { "fileName", n.Name }},
new AjaxOptions { HttpMethod = "GET", UpdateTargetId = "file-view" })
```

Obr. č. 11 Ukážka pomocnej AJAX metódy

Testovanie
Testovanie bolo v tomto prípade priamočiare. Bolo potrebné overiť, či po kliknutí na názov súboru sa upraví zobrazovaný dokument. Zobrazované súbory boli vo finálnej verzii vždy načítané a zobrazené korektne.

5.5 ZOBRAZENIE GRAFU ODOVZDANÍ PROJEKTU

Graf odovzdaní projektu má slúžiť používateľovi ako prehľad o odovzdaniach („changesetoch“) projektu. Jednotlivé odovzdania v grafe predstavujú body grafu, ktorý sa podľa vlastností projektu môže rozvetvovať. Vetvenie môže spôsobiť prípad, keď z jednej verzie projektu vzniknú dve paralelné. Na úrovni bodov grafu je umiestnená informácia o jednotlivých verziách.

Analýza
Keďže sa má graf zobrazovať na web stránke, ďalej budú analyzované možnosti zobrazenie grafu v jazyku JavaScript prípadne jeho rozšírenia JQuery.

Existuje mnoho knižníc na zobrazovanie grafových štruktúr a stromov, ktoré zobrazujú obsah pomocou HTML5 elementu canvas. Canvas slúži na zobrazovanie grafických údajov na webe. Obsahuje teda grafickú knižnicu, ktoré ponúka funkcie na vytvorenie grafického obsahu.

Analyzované boli niektoré JavaScript knižnice, ktoré využívali HTML5 canvas:

- Sigma.js
- Processing.js
- jit.js
- D3.js

Údaje pre graf je nutné načítať zo systému AST-RCS, tieto údaje by mali obsahovať informácie o jednotlivých odovzdaniach („changesetoch“) projektu. Vzorom k tejto operácií bol analyzovaný príbeh - **Zobrazenie zoznamu odovzdaní projektu.** Tento príbeh poskytuje základné funkcie k načítaniu potrebných údajov. Tieto údaje je potrebné zotriediť do stromu tak aby mohol byť zobrazený.

Návrh
Po analýze jednotlivých knižníc sa ako najefektívnejšia možnosť ukázala nepoužiť ani jednu z nich a využiť základné funkcie HTML5 canvas:

Znazornenie bodu:

```
ctx.arc(x, y, 10, 0, 2 * Math.PI);
ctx.fill();
```

Znázornenie spojenia:

```
ctx.moveTo(x, y);
ctx.lineTo(xx, yy);
```

Načítanie dát do stromu môže byť uskutočnené iteráciou poľa a vyplnenie potomkov jednotlivých entít.

Riešenie
Jednotlivé načítané „changesety“ na zobrazenie boli usporiadané do stromu pomocou triedy:

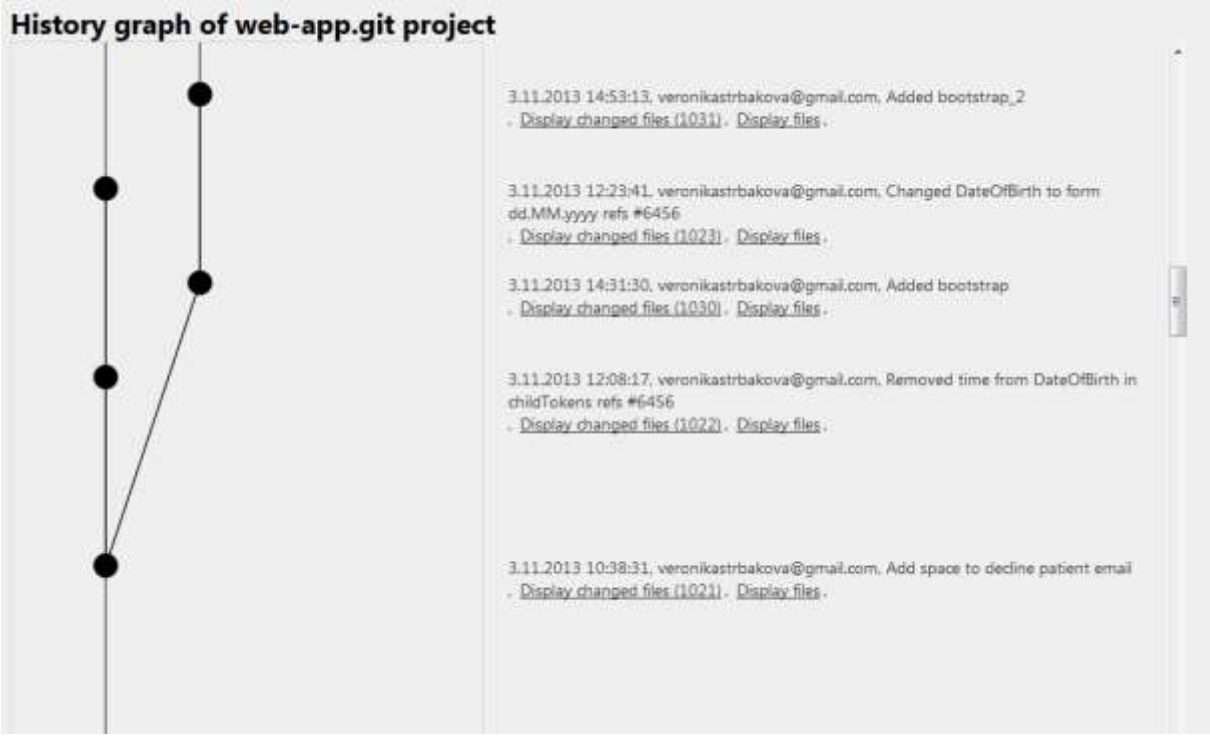
```
public class ChangesetModel
{
        public int ID { get; set; }
        public string IdInRcs { get; set; }
        public int? AncestorID1 { get; set; }
        public int? AncestorID2 { get; set; }
        public int CommiterID { get; set; }
        public string CommiterLogin { get; set; }
        public string TimeStamp { get; set; }
        public string Message { get; set; }
        public int ProjId { get; set; }
        public int?[] Children = new int?[2] { null, null };
}
```

Model bol prenesený a zobrazený na používateľovu stranu do formátu JSON vďaka funkcií:

```
var model = @Html.Raw(Json.Encode(Model));
```

Táto transformácia je zadefinovaná v príslušnom „View“ , ktorý zobrazuje stránku s grafom. Po tom ako sa načíta stránka sa spustí algoritmus vykresľovania grafu v jazyku JavaScript. Algoritmus rekurzívne prechádza vytvorený strom a podľa vetvenia rozmiestňuje body tak aby ku každému na jeho pravej strane prislúchal daný popis.

Canvas sa rozširuje spolu s rozširovaním grafu a je možné skrolovať v rámci takto vytvoreného grafu, vďaka vnoreniu elementu „canvas“ do elementu „div“.



Obr. č. 12 Ukážka grafu zmien v projekte

Testovanie

Na testovanie tohto príbehu je vytvorený automatický jednotkový test, ktorý sa spúšťa spolu s nasadzovaním kódu.

5.6 ZOBRAZENIE ZOZNAMU ZMIEN VYKONANÝCH VO ZVOLENOM ODOVZDANÍ

Pri vytváraní projektu vznikajú stále novšie verzie projektu. V týchto verziách je niekedy potrebné zobraziť zmeny, ktoré v zvolenom odovzdaní nastali. V tejto požiadavke budeme vytvárať metódu pre zobrazenie zmien vo zvolenom odovzdaní oproti predchádzajúcej verzii.

Analýza

Analyzovali sme prístupy k zisťovaniu rozdielov v zdrojových kódoch. Nástroj Diff slúži na zobrazenie rozdielov dvoch súborov. Pre jazyk C# existuje knižnica DiffModel, ktorá umožňuje porovnanie súborov s rôznym výstupným zobrazením.

Návrh

Pomocou knižnice DiffModel implementujeme metódu porovnávania súborov. Podľa výstupu označíme pridané, zmazané a zmenené riadky v zdrojovom kóde. Označené riadky zobrazíme v používateľskom rozhraní.

Riešenie

Implementovali sme metódu, ktorá pomocou knižnice DiffModel zistí rozdiely v dvoch verziách zdrojových kódov súborových entít.

```
var EntityDifferences = changesInSelectedCommit(changesetId);
```

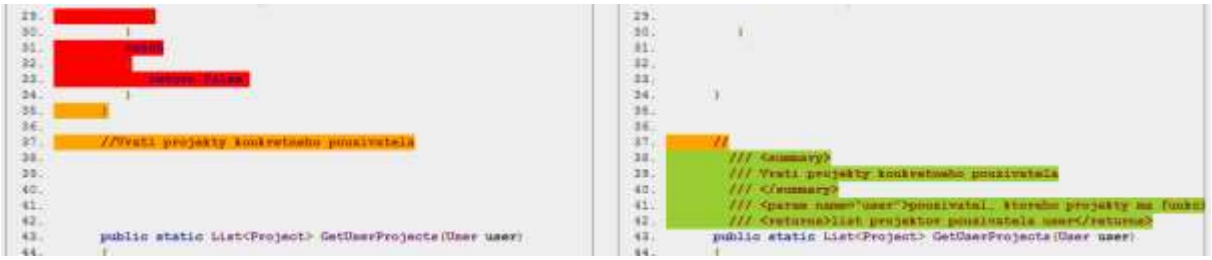
Metóda označuje riadky v kóde, na ktorých nastala zmena. V prípade zmazania súboru alebo vzniku nového súboru sa zmeny nezisťujú.

```
var d = new Differ();

var inlineBuilder = new SideBySideDiffBuilder(d);
var result = inlineBuilder.BuildDiffModel(oldFileContent.Content, newFileContent.Content);
foreach (var line in result.OldText.Lines)
{
        actualLine = "";
        switch (line.Type)
        {
                case DiffPlex.DiffBuilder.Model.ChangeType.Inserted:
                        actualLine += "+";
                        break;
                case DiffPlex.DiffBuilder.Model.ChangeType.Deleted:
                        actualLine += "-";
                        break;
                case DiffPlex.DiffBuilder.Model.ChangeType.Modified:
                        actualLine += "*";
                        break;
        }
        actualLine += line.Text + '\n';
        oldFileChanges += actualLine;
}
```

Program prečíta všetky riadky starej verzie súboru a označí zmeny. Rovnaký postup opakuje s novou verziou súboru.

Zobrazenie v používateľskom rozhraní obsahuje porovnané zdrojové kódy s farebným označením zmenených riadkov. V prípade zmazaného alebo novo vzniknutého súboru sa zobrazuje iba zdrojový kód súboru.



Obr. č. 13 Príklad zobrazenia zmien v súbore

Testovanie

Pri testovaní sme porovnávali nami zobrazené zmeny v našom projekte so zobrazenými zmenami v systéme TFS.

5.7 ZOBRAZENIE ZDROJOVÉHO KÓDU ZMENENEJ ENTITY V ODOVZDANÍ SO ZVÝRAZNENÝMI ZMENAMI

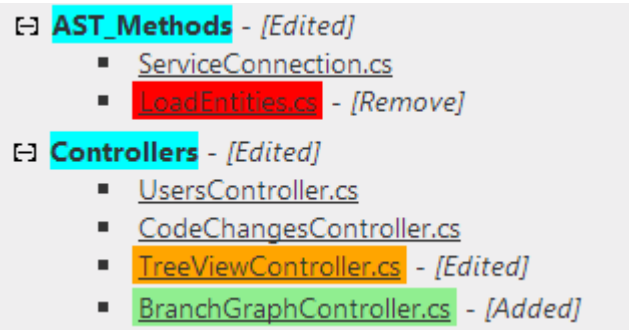
Zobrazenie zdrojového kódu zmenenej entity vo zvolenom odovzdaní. Úlohou bolo graficky zobraziť zmeny v stromovej štruktúre projektu zo zvoleného odovzdania z predchádzajúceho stavu.

Riešenie

Farebne odlíšené zmeny v graficky zobrazenej stromovej štruktúre projektu. Súbory sa zobrazujú s príznakmi a farebne rozlíšené podľa typu zmeny. Adresáre, v ktorých sú súbory uložené, farebne rozlišujeme spolu s príznakom (napr. *[Edited]*). Rozlišujeme tieto tri zmeny:

Farba	Príznak	Popis
Oranžová	[Edited]	Upravený súbor
Zelená	[Added]	Pridaný nový súbor
Červená	[Remove]	Vymazaný súbor

Pre zobrazenie vykonaných zmien (všetky 3 typy zmeny súboru) v súboroch adresára, je adresár rozlíšený bledo modrou farbou.



Obr. č. 14 Farebne rozlíšené zobrazenie súborov

5.8 NAČÍTANIE HISTÓRIE SÚBOROVEJ ENTITY Z AST-RCS

Používateľ potrebuje vedieť informácie o danej súborovej entite. Kto na nej pracoval, kedy na nej vykonával zmeny a opis toho, aké zmeny na danej súborovej entite vykonal.

Návrh

Údaje je potrebné načítať z AST-RCS na základe Id súborovej entity. Tieto údaje sa uložia do modelu.

Riešenie

Tieto údaje načítame z AST RCS pomocou

```
var projectsets = c.GetFileChangesets(new GetFileChangesetsRequest()
	{
		EntityId = entityId
	});
```

Kde `c` je `AstRcsClient` Tieto údaje sú následne uložené do modelu, ktorý je zobrazovaný používateľovi .

Testovanie

V tejto fáze testovanie nebolo riešené pomocou automatických naprogramovaných testov. Výsledky boli testované pre známu súborovú entitu.

5.9 NAČÍTANIE HISTÓRIE ENTITY ZDROJOVÉHO KÓDU Z AST-RCS

Našou úlohou je zistiť a vypísať, kedy nastali zmeny v kódovej entite, ktorou môže byť napríklad trieda alebo metóda, čiže nejaký menší prvok súboru. Tieto zmeny získame ako zoznam odovzdaní projektu – changesetov, kedy bola daná entita nejako upravovaná.

Analýza

Prvou otázkou, ktorá sa vyskytla pri analýze entity v AST-RCS je určenie jej ID. Entita môže mať 2 druhy ID:

- VersionID – ID konkrétnej inštancie entity (verzie)
- EntityID – ID, pomocou ktorého je identifikovaná entita bez ohľadu na jej verziu, čiže akoby nejaké globálne ID entity v projekte

Pri spracovaní určitej entity bude vhodnejšie ju najprv identifikovať podľa *VersionID*. Pomocou tohto čísla je potom možné získať *EntityID*. S ním dokážeme vytiahnuť z AST-RCS pomocou použitia správnej služby zoznam changesetov, kedy bola menená.

Návrh

Pri volaní metódy na získanie histórie entity odovzdáme parameter *VersionID*. Pomocou neho vyhľadáme konkrétnu entitu a jej changesety. Tie zaznamenáme do novovytvoreného modelu a následne pošleme do View, ktoré bude slúžiť na vhodný výpis histórie entity.

Riešenie

Pre implementáciu sme vytvorili metódu *CodeEntityHistory* nachádzajúcu sa v súbore *CodeChangesController.cs*.

Prvým krokom bolo klasicky vytvorenie klienta umožňujúceho prístup k službám v AST-RCS. Pomocou služby *GetCodeEntity* a parametra *VersionID* sme získali popis konkrétnej metódy, ktorý obsahuje aj názov, typ a *IdentityID* entity. Tieto údaje sme zaznamenali do modelu. Model obsahuje názov, typ entity a zoznam changesetov, kedy bola entita menená. Z informácií o entite sme však ešte nezískali konkrétne changesety. Na to sme použili službu GetCodeEntityChangeset. Získaným zoznamom *changesetov* sme naplnili list v modeli. Takto sme dospeli k naplneniu nášho modelu a môžeme ho ďalej poslať na zobrazenie. Na to nadväzuje User Story s názvom Zobrazenie histórie entity vo forme tabuľky.

Testovanie

Pri testovaní sme skúmali, či sa správne zobrazí výsledok a či existuje ošetrenie, ak by sa entita s daným *VersionID* nenašla. Napísali sme test, ktorý overuje, či bolo po zavolaní našej metódy následne zobrazené View.

5.10 Zobrazenie histórie entity vo forme tabuľky

Ako už názov napovedá, jedno z vhodných zobrazení histórie entity je formou tabuľky. Preto vytvoríme také View, ktoré bude spĺňať naše požiadavky.

Analýza

Ako vstup dostávame model, ktorý obsahuje meno, typ a zoznam changesetov pre danú entitu. Tento musíme spracovať a vhodne vypísať. Na tvorbu tabuľky použijeme HTML a CSS kód.

Riešenie

V súbore *EntityHistory.cshtml* sme si definovali HTML kód pre zobrazenie tabuľky. Jej vlastnosti sú doplnené v CSS súbore *site.css*. Tým sa zobrazí prehľadná tabuľka, s vyznačenými čiarami a formátovaním. Tá obsahuje výpis listu changesetov. V modeli, ktorý bol poslaný do View, sme mali ešte názov a typ entity, ktoré sme vypísali nad tabuľku pre doplnenie informácií.

Testovanie

Testovanie prebiehalo hlavne vizuálne, či daná tabuľka spĺňa v dostatočnej miere jednoduchý a prehľadný graficky dizajn. Ako sme už spomínali pri príbehu Načítanie histórie entity zdrojového kódu z AST-RCS, bol napísaný test ktorý kontroluje či sa zobrazilo toto View. Ošetrili sme prípad, že ak prišiel na vstup prázdny model, oznámili sme používateľovi, že zadal nevhodný parameter.

5.11 Porovnanie zdrojového kódu dvoch zvolených verzií entity z AST-RCS

V tomto príbehu bolo potrebné umožniť porovnanie zdrojového kódu dvoch zvolených verzií entity.

Bolo potrebné umožniť používateľovi systému prezeranie históriu verzií zvolenej kódovej entity tým, že si ľubovoľne zvolí dve verzie danej entity, ktorých obsah chce porovnať a následne sa mu zobrazia vedľa seba so zvýraznenými zmenami.

Analýza

Bola analyzovaná možnosť získania histórie verzií zvolenej entity z systému AST-RCS. Na základe týchto poznatkov, bol vytvorený návrh pre riešenie príbehu.

Návrh

Na riešenie tohto príbehu je potrebné vytvoriť kontrolér, ktorý spracuje údaje o verziach zvolenej entity. Údaje o verziách entity je vhodné ukladať, preto je vhodné vytvoriť si model, ktorý obsahuje údaje ako, *versionId* (id verzie entity)*, commiter* (osoba, ktorý pridal entity do repozitára)*, commitId* (id odovzdania do repozitára) a *date* (dátum odovzdania). Tieto údaje sa majú zobraziť aj pri prezerania histórií zvolenej entity. Na zvýraznenie zmien sa má použiť knižnica *DiffBuilder.*

Riešenie

Počas implementácie bol vytvorený kontrolér *EntityVersionController.* Kontrollér obsahuje dve základné metódy. Metódy kontroléra:

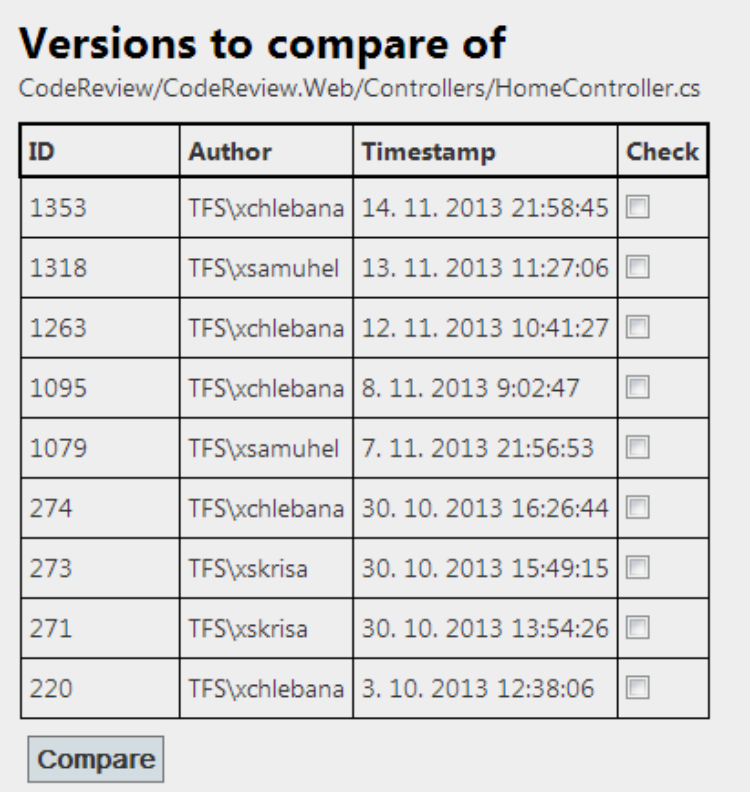
- *Index()* - na základe *versionId* získa históriu zvolenej verzie, údaje o verziách sú ukladané v modeli *EntityVersionModel* (obrázok č. 15)
- *FilesCompare()* – porovnáva dve zvolené verzie entity, zmeny v dvoch verziách sa zvýraznia

```
public class EntityVersionModel
    {
        public List<EntityVersion> entityVersionList { get; set; }
    }

    public class EntityVersion
    {
        public int commitId { get; set; }
        public int versionId { get; set; }
        public string commiter { get; set; }
        public DateTime date { get; set; }
    }
```

Obr. č. 15 *EntityVersionModel*

Naplnení model je ďalej reprezentovaná vo príslušnom View, kde sa zobrazí história verzií zvolenej entity s možnosťou výberu dvoch na porovnanie(Obrázok č. 16).



Obr. č. 16 Zobrazenie histórií verzie zvolenej entity

5.12 Načítanie súborovej štruktúry projektu do stromovej reprezentácie pre novú verziu AST-RCS

Nová verzia AST-RCS obsahuje zmenené prístupy k načítaniu údajov z databázy. Podľa novej dokumentácie sa metódy načítania údajov upravia.

Riešenie

V novej verzií AST-RCS bolo potrebné načítať najprv všetky changesety(zoznam zmien). Podľa zvoleného id changesetu bolo následné možné načítať súbory zvolenej verzie projektu. Do uzlu v strome pribudli údaje o VersionId (id verzie) súboru. Podľa tohto ID je následné možné zobraziť detaily zvolenej verzie súboru.

5.13 Načítanie AST štruktúry projektu do stromovej reprezentácie pre novú verziu AST-RCS

Nová verzia AST-RCS priniesla zmeny v metódach, ktoré zasiahli príbeh: Načítanie AST štruktúry projektu do stromovej reprezentácie. Preto bolo nutné pozmeniť štruktúru načítavania údajov z tohto systému.

Analýza

Zmeny v systéme zasiahli hlavne správu verzií kódových entít. Preto je v novej verzií nutné v prvom rade zadefinovať, z ktorého „changesetu“ alebo verzie chce systém načítať kódové entity. Nová verzia použitej metódy *SearchCodeEntity()* už neumožňuje načítať kódové entity podľa identifikátoru projektu, ale práve podľa „changesetu“.

Cieľom tejto metódy bolo načítať kódové entity poslednej verzie daného projektu. Na načítanie „changesetov“ služia v novej verzií metódy:

SearchChangesets() – vráti zoznam „changesetov“, ktoré vyhovujú kritériám

GetChangesets() – vráti údaje o danom „changesete“ podľa identifikátoru

Návrh

Keďže vstupom do metódy ostáva identifikátor daného projektu, je treba načítať identifikátor posledného „changesetu“ tak, aby boli načítané najnovšie kódové entity. Tento identifikátor je už možné použiť v rovnakých metódach, ktoré boli použité v prvej verzií.

Riešenie

Vo finálnom riešení nenastala vážnejšia zmena. Zmena nastala iba v načítavaní údajov zo systému AST-RCS.

Testovanie

Testovanie je prevedené rovnako ako v prechádzajúcej verzií.

5.14 Zobrazenie zdrojového kódu zvolenej entity z novej verzie AST-RCS

Úlohou bola zmeniť už vytvorené funkcionality nášho systému pre prácu s novou verziou systému AST-RCS.

Riešenie

Zmeny boli vykonané v kontroléri *FileRequestController*. Opravili sa volania služieb AST-RCS na volania nových služieb. Nebola pôsobená výrazná zmena funkcionality spomenutého kontroléra.

5.15 Zobrazenie zoznamu odovzdaní projektu v novej verzii AST-RCS

Po prvom šprinte sa zmenila štruktúra služieb v AST-RCS. Preto je nevyhnutná úprava doterajších riešení. Našou úlohou je upraviť získavanie zoznamu odovzdaní projektu.

Analýza

Hlavným rozdielom voči doterajšiemu riešeniu je, že všetky potrebné služby sú aktuálne pohromade a nepotrebujeme vytvárať viacero klientov. S pribúdajúcimi changesetmi a testovaním sme si tiež všimli, že je potrebné ošetriť stránkovanie, pretože naraz sa dá získať iba 100 changesetov. Novinkou je tiež vyhľadanie changesetov podľa ID projektu, ktoré nám v predchádzajúcej verzii chýbalo.

Riešenie

Pri implementácií sme upravili program podľa analyzovaných nedostatkov a zmien.

V prvom rade sme použili iba jedného klienta na pripojenie k serveru a zodpovedajúcim službám. Po pripojení sme vyhľadali changesety, pričom sme zadali parameter, ktorým je ID projektu a nie čas vytvorenia, ako to bolo predtým. Ako výsledok sme nedostali zoznam ID changesetov ale už pole changesetov zložených z prvkov *ChangesetDto*. V tomto modeli nastala tiež zmena. Nepotrebujeme už zisťovať login používateľa a hľadať ho osobitne, model changesetu obsahuje prvok *Commiter* s potrebnými informáciami. Preto sme upravili aj náš model, do ktorého ukladáme changesety. Skladá sa z podobnej štruktúry ako model ChangesetDto na serveri, avšak informácie si kopírujeme do vlastného modelu. Je to z dôvodu, že ak by nastala niekedy v budúcnosti opäť zmena na serveri, tak nemusíme byť viazaný na konkrétny serverový formát, ale máme vlastný. Informácie tak uložíme do listu pozostávajúceho z nášho modelu a ten posielame do View, kde sa zobrazí v prehľadnej tabuľke. Poslednou doplnenou funkcionalitou je stránkovanie. To je pridané z dôvodu, že môžeme získať maximálne 100 changesetov na jedno hľadanie na serveri, preto ho musíme urobiť viac krát podľa počtu stránok.

Testovanie

Testovanie prebiehalo ako v predošlej verzii, pomocou TFS servera. Rozdielom však je filtrovanie podľa ID projektu, čiže pri zadaní ID nášho projektu výpis presne zodpovedal výpisu na našom serveri. Stránkovaním sme zabezpečili, že sa zobrazili aj posledné vykonané changesety, ktoré sa pri testovaní bez neho nezobrazovali, keďže ich počet prekročil 100.

5.16 Úspešné prihlásenie s neplatným heslom

Metóda, ktorá sa stará o prihlasovanie obsahovala chybu, ktorá umožnila používateľovi prihlásiť sa s neplatným heslom.

Riešenie

Metódu, ktorá kontrolovala, či používateľ existuje v systéme mala návratovú hodnotu existujúceho používateľa aj keď nezadal správne prihlasovacie meno. Riešením bolo vrátenie hodnoty *NULL.*

6 STONKA

Číslo šprintu: 3

Začiatok šprintu: 6.11.2013

Koniec šprintu: 20.11.2013

Príbehy:

- **Vytvorenie vizuálu stránky**
- **Vytvorenie kontextového menu pre položky**
- **Kontrola prístupu k dátam**
- **Filtrácia histórie odovzdaní**
- **Stránkovanie histórie odovzdaní**
- **Priradenie projektov k používateľovi**
- **Priradenie používateľov k projektu**
- **Určenie administrátorov systému**
- **Definovanie aktívnych odkazov v kontextovom menu**
- **Úplné zobrazenie číslovania riadkov**
- **Vizualizácia zmeny v strome**
- **Vizualizácia rozbalenia uzla stromu**
- **Vytvorenie atribútov na autentifikáciu pre vytvorený model**
- **Prázdne riadky na začiatku a konci kódu**

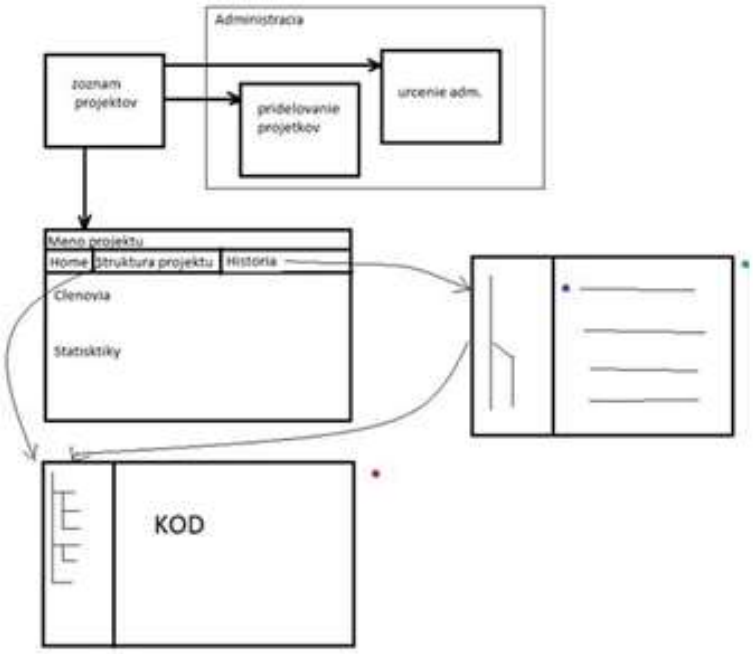
6.1 VYTVORENIE VIZUÁLU STRÁNKY

Je potrebné vytvoriť jednotné používateľské rozhranie, v ktorom sa môže používateľ jednoducho orientovať. Na navigáciu je potrebné vytvoriť menu projektu, ktoré bude obsahovať kategorizované odkazy na jednotlivé časti zobrazenia projektu.

Analýza

Analyzovali sme prístupy vytvárania jednoduchého rozbaľovacieho menu pomocou HTML a CSS.

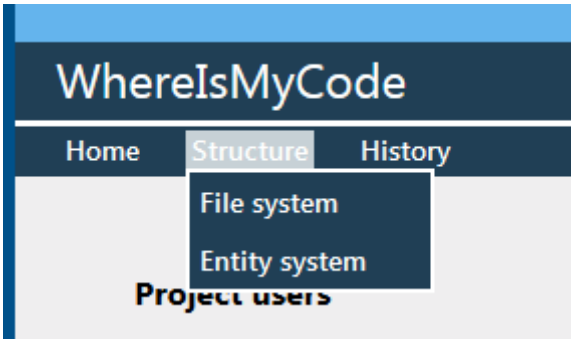
Návrh



Obr. č. 17 Návrh navigácie na stránke

Navrhli sme zobrazenie stránky projektu. Menu projektu je umiestnené v hornej časti stránky. Jednotlivé odkazy menu majú prepojenie na zobrazenie konkrétnych komponentov.

Riešenie



Obr. č. 3 Zobrazenie na stránke

V jazyku HTML a pomocou štýlov CSS sme vytvorili jednoduché používateľské menu. Jednotlivé položky menu:

- Home – informácie o zvolenom projekte, zoznam používateľom, ktorým je projekt pridelený, neskôr pribudnú štatistické údaje

- Structure – obsahuje štruktúru projektu
    - File system – strom súborov v projekte
    - Entity system – strom jednotlivých entít v projekte
- History – zobrazenie histórie zmien v projekte

6.2 Vytvorenie kontextového menu pre položky

Cieľom tohto príbehu je vytvorenie jednoduchého kontextového menu pre položky v strome, ktoré nám poskytne ďalšie možnosti, akými s nimi môžeme pracovať.

Analýza

V tomto príbehu je potrebné zistiť, akým spôsobom budeme reprezentovať samotné menu, kde v dokumente sa bude nachádzať, akým spôsobom zabezpečíme vizuálnu úpravu dokumentu a zobrazovanie menu, a zároveň ako to bude celé vyzerať.

Návrh

Návrh riešenia sa skladá z troch hlavných častí.

- HTML kód - Bolo potrebné upraviť HTML kód a ku každému uzlu pridať prvok, ktorý bude predstavovať jednoduché kontextové menu. Toto menu musí byť zároveň jednoducho editovateľné, to znamená, že jeho obsah musí byť definovaný tiež v tomto dokumente.
- JavaScriptový kód - Bolo potrebné pridať kód, ktorý po kliknutí na tlačidlo zobrazí toto menu a pri opätovnom kliknutí naňho toto menu skryje. Taktiež musia byť pokryté situácie, kedy je menu zobrazené a používateľ klikne na iné tlačidlo pri inej položke prípadne klikne niekde inam v rámci dokumentu.
- CSS kód - Bolo potrebné menu naštýlovať.

Riešenie

Samotné riešenie pozostáva s trochu spomínaných častí. V HTML sa nachádza menu, ktoré obsahuje zvolené položky. Toto menu je pri načítaní dokumentu neviditeľné. O to sa stará CSS kód, ktorý definuje, že prvok nie je zobrazený. Po kliknutí na ikonku pre menu sa zobrazí príslušné menu pre danú položku v strome. Menu je zobrazené hneď vedľa ikonky a takéto zobrazenie je zabezpečené nastavením absolútnej pozície v rámci dokumentu. Samotná pozícia je vyrátaná práve na základe pozície tlačidla (obr. č. 18).

```
menu.css({ 'top': $(this).offset().top, 'left': $(this).offset().left +
$(this).outerWidth() })
```

Obr. č. 18 Výpočet a úprava pozície

Testovanie

Testovanie prebiehalo overovaním očakávaného správania a overovaním všetkých situácii popísaných v časti "Návrh". Testovanie prebehlo úspešne.

6.3 Kontrola prístupu k dátam

Používateľ požaduje zabezpečenie prístupu k dátam tak, aby mal každý používateľ právo vidieť dáta len z projektu, ku ktorému je priradený.

Analýza

Ak nekontrolujeme prístup k dátam na stránke, môže nastať situácia, že používateľ zmení údaje v odkaze (v linku) a zobrazia sa mu dáta, ku ktorým možno nemá prístup. V našom prípade môže

používateľ prezerať obsah ľubovoľného projektu, ak zmení *Id* projektu. Z tohto dôvodu musíme kontrolovať tento prístup voči právam používateľa.

Tento problém sa dá riešiť rôznymi spôsobmi, zvažovali sme nasledujúce možnosti:

A) Vytvorenie metódy na správu prístupu, ktorá sa volá na začiatku controllera, ktorý by teoreticky mohol zobraziť používateľovi dáta ku ktorým nemá prístup.
   Nevýhody:
   - neprehľadnosť kódu
   - duplicita kódu

B) Vytvorenie atribútov. Táto metóda je efektívna prehľadná. Použitie je jednoduché – nad definíciu controllera sa napíše atribút do hranatých zátvoriek („*[]*“). Atribúty sú jazykové konštrukcie, ktoré môžu doplniť elementy programového kódu o špecifické doplňujúce informácie.

Na prístup k súborom sme museli zistiť *id* projektu podľa *id* súboru. *AST-RCS* neposkytovalo takúto možnosť, preto sme museli požiadať *GRATEX* o implementovanie tejto funkcionality.

Riešenie

Pre tento problém sme zvolili riešenie pomocou atribútov. V súčasnom stave sme potrebovali vyriešiť kontrolu prístupu k projektu a k súboru.

Kontrola prístupu k projektu:

Vytvorili sme atribúty `[AccessDeniedProject]` a `[AccessDeniedChangeset]`. Tieto atribúty sa píšu pred definíciu controllera, ktorý obsahuje parameter *projectId* a pre súbor *changesetId*.

```
[AccessDeniedProject]
public ActionResult Index(int projectId = 0, string projectName = "")
{
```

Obr. č. 19 Použitie atribútu *AccessDeniedProject*

```
[AccessDeniedChangeset]
public ActionResult TreeViewChanges(int type = 0, int? changesetId = null)
{
```

Obr. č. 20 Použitie atribútu *AccessDeniedChangeset*

Atribút *AccessDeniedProject* nájde podľa parametra v linku *projectId* a overí či aktuálne prihlásený používateľ má prístup k tomuto projektu. Ak používateľ má prístup, controller pokračuje vo vykonávaní. Ak nemá, zobrazí sa chybová stránka o zamietnutí prístupu.

```
public class AccessDeniedProjectAttribute : AuthorizeAttribute
{
  public override void OnAuthorization(AuthorizationContext filterContext)
  {
    if (filterContext.HttpContext.Request.Params["projectId"] != null &&
        !CrAuthorize.CanViewProject(UserMethods.GetUserIdFromName(
          filterContext.HttpContext.Request.Params["AUTH_USER"]),
          int.Parse(filterContext.HttpContext.Request.Params["projectId"])))
          {
                    filterContext.Result = new ViewResult() { ViewName = "AccessError" };
          }
  }
}
```

Pre atribút *AccessDeniedChangeset* je prístup analogický ako pri *AccessDeniedProject* s rozdielom, že sa pozerá na parameter *changesetId*.

6.4 FILTRÁCIA HISTÓRIE ODOVZDANÍ

V projekte je implementovaná funkcionalita, ktorá nám zobrazí v grafe odovzdania projektu. Bola vytvorená v druhom šprinte s názvom Korienok, pod príbehom s názvom Zobrazenie grafu odovzdaní projektu. Našou úlohou je upraviť túto funkcionalitu tak, aby nám poskytla iba uzly a graf changesetov, kde sa vyskytuje konkrétna nami zvolená entita.

Analýza

V prvom rade sme analyzovali naprogramované riešenie príbehu Zobrazenie grafu odovzdaní projektu. To poskytuje výpis všetkých changesetov projektu v podobe grafu. Niektoré uzly v grafe však potrebujeme vynechať. Tu využijeme funkcionalitu načítania histórie entity, ktorá už bola vypracovaná v príbehoch Načítanie histórie entity zdrojového kódu z AST-RCS a Načítanie histórie súborovej entity z AST-RCS. Pomocou nich získame changesety, ktoré sa týkajú histórie určitej entity, čiže uzly, ktoré máme ponechať zo všetkých uzlov projektov. Našou úlohou je teda odstrániť nepotrebné uzly a správne pospájať tie, ktoré nám ostanú, čiže vytvoriť filter.

Návrh

Správnym postupom je využitie už naprogramovaných metód pre získanie potrebných changesetov, čiže uzlov grafu. Z naprogramovaných metód tak dostaneme 2 potrebné listy a tými je list changesetov projektu a changesetov entity.

V ďalšom kroku potrebujeme aplikovať filter. Tie changesety, ktoré sa nenachádzajú v liste changesetov entity môžeme vymazať. Musíme však dávať pozor na to, aby sme správne opravili a vyplnili predchodcov changesetov. Tu nastáva viacero prípadov, ktoré bude potrebné ošetriť.

Po správnom vymazaní changesetov dostaneme iba tie, ktoré potrebujeme vypísať. Pred zobrazením však ešte musíme opraviť zoznamy children v jednotlivých changesetoch. Po ich úprave vytvoríme model, ktorý následne dokáže spracovať JavaScript pre zobrazenie grafu. Výsledný model pošleme do View, ktoré zobrazuje graf odovzdaní projektov.

Riešenie

V prvom rade musíme oznámiť, že riešenie implementované v tomto šprinte nie je konečné a bude potrebné na ňom ďalej zapracovať. Pri práci riešiteľ neodhadol správne svoj časový plán a stihol vytvoriť iba základné riešenie.

Aktuálny výsledok poskytuje úpravu takého typu projektu, ktorý nie je rozvetvený. Taktiež zatiaľ spracujeme len históriu changesetov pre súborovú entitu. Metóda je dočasne vytvorená z časti iných metód, hlavne kvôli jej vývojovému charakteru. Momentálne sme sa však zamerali na reprezentovanie základnej funkcionality tohto príbehu s tým, že celkovú funkcionalitu čo najskôr dopracujeme počas ďalšieho šprintu.

Metóda pre zobrazenie histórie súborovej entity v grafe (`FileEntityBranchGraph`), ktorá je aktuálnym výsledkom, sa nachádza v súbore *BranchGraphController.cs*. Po jej dokončení a otestovaní z nej vezmeme a použijeme iba určitú časť, ktorá poslúži ako filter.

Pri vývoji metódy sme použili v prvom kroku funkciu pre získanie changesetov (*LoadEntities.LoadChangesetsToTree*). Druhým krokom bolo získanie changesetov pre nami zadanú

súborovú entitu s VersionID zadaným v parametri. Počas tohto načítania sme vytvorili list, ktorý obsahuje ID changesetov ktoré nemáme vymazať.

Po načítani oboch listov changesetov sme prechádzali changestmi projektu a zisťovali, či je jednotlivý changeset určený na vymazanie alebo ho máme ponechať. Ak sme ho mali ponechať, pokračovali sme ďalej. Ak je určený na vymazanie, hľadali sme iné changesety, ktorým je predkom. V nich sme potom upravili ID changesetu predka tak, aby sme changeset určený na mazanie vynechali. Zaznamenali sme si, ktorý changeset treba vymazať. Po prejdení všetkých changesetov a úprave predkov sme vymazali nepotrebné changesety. Ostali iba tie, ktoré prešli naším filtrom, pričom im boli upravený predkovia. Predtým, ako sme tento zoznam zobrazili v grafe sme ešte upravili zoznamy detí v jednotlivých changesetoch. Je to potrebné pre JavaScript a korektné zobrazenie v grafe. Správne vyplnený *modelBranchGraph* následne posielame do už vytvoreného View pre históriu projektu a zobrazíme prefiltrovaný graf.

Testovanie

Testovanie zatiaľ prebiehalo iba vizuálnou kontrolou. Zistili sme však, že nefunguje odkaz na porovnanie posledných verzií changesetov, čo je zrejme spôsobené úpravou predkov, ktorú sme vykonali. Taktiež sme si všimli, že text changesetov sa prelína. Tento problém je už zaznamenaný v product backlogu a bude sa riešiť.

6.5 Stránkovanie histórie odovzdaní

Kvôli možnosti ľubovoľnej veľkosti grafu je možné, že canvas dosiahne maximálnu veľkosť. V tomto prípade používateľ neuvidí kompletný graf. Preto je nutné nájsť nový spôsob zobrazenia dát alebo stránkovania, ktorý zabráni dosiahnutiu maximálnej veľkosti canvasu.

Analýza

Vytvorenie stránkovania je založené na statickej veľkosti elementu canvas a zobrazovania len rozsah viditeľný používateľom.

Ovládanie stránkovania je možné vytvorením nového „scroll baru“, ktorý bude ovládať vykresľovanie v statickom canvase. Avšak je tiež možné využiť existujúci „scroll bar“, ktorý v súčasnej verzií ovláda skrolovanie elementu „div“, v ktorom sú vnorené elementy canvas a element, ktorý obsahuje popisi k jednotlivým bodom grafu.

Návrh

Menej zložitou metódou je zanechať „scroll bar“, ktorý bude ovládať iba časť s popismi. Samotný canvas bude mať fixnú pozíciu, avšak jeho obsah sa na základe posunu „scroll baru“ bude meniť. To bude vytvárať ilúziu skrolovania, v skutočnosti sa však bude iba meniť obsah canvasu. Pri takomto prístupe je možné zobraziť graf s ľubovoľnou veľkosťou.

Zobrazovanie iba viditeľnej časti údajov je možne dosiahnuť metódou, ktorá pri načítaní stránky vypočíta všetky pozície bodov v grafe. Pri skrolovaní sa zistí poloha a podľa nej je možné vybrať ten správny obsah.

Riešenie

Úprava oproti predošlej verzií spočíva v úprave výpočtovej funkcií, ktorá predpočíta obsah a do štruktúry uloží koordináty každého bodu.

Pri každom pohybe „scroll baru“ za zavolá funkcia:

```
$("#BranchGraphContainer").scroll(function () {
```

```
        ctx.clearRect(0, 0, 400, 600);
        drawnode($("#BranchGraphContainer").scrollTop());
    });
```

Táto funkcia vymaže predošlí zobrazený obsah a pomocou funkcie *drawnode(x)* sa vykreslí práve obsah od koordinátu x, ktorý reprezentuje polohu „scroll baru“.

Testovanie

Testovanie je prevedené rovnako ako v prechádzajúcej verzií príbehu: ZOBRAZENIE GRAFU ODOVZDANÍ PROJEKTU.

6.6 PRIRADENIE PROJEKTOV K POUŽÍVATEĽOVI

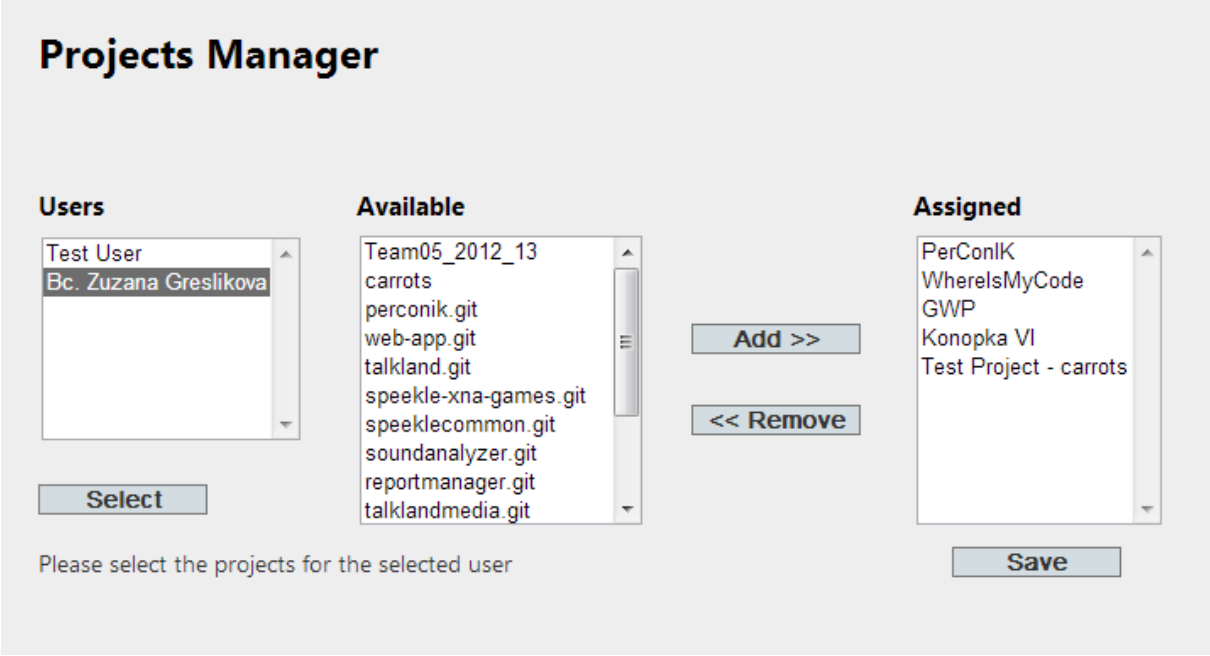
Tento príbeh nadväzuje na príbeh z prvého šprintu 4.4. Niekedy je nevyhnutné pridať jednému používateľovi väčšie množstvo projektov. Je dôležité, aby rozhranie bolo čo najjednoduchšie a aby administrátor nemusel každý projekt priradiť používateľovi individuálne, a aby bolo možné zvoliť skupinu projektov, ktoré budú používateľovi priradené.

Návrh

Je potrebné vytvoriť rozhranie ktoré umožní multiselekciu projektov.

Riešenie

Vytvorí sa model, do ktorého sa vloží zoznam všetkých používateľov. Ako prvý sa zobrazí prvý používateľ zo zoznamu, a preto model obsahuje ďalej list projektov priradených tomuto používateľovi a taktiež list projektov, ktoré mu nie sú priradené. Model sa zobrazí ako je možné vidieť na obrázku č. 21, pričom „Users“ predstavuje zoznam používateľov (so zvoleným prvým používateľom). „Available“ predstavuje projekty, ku ktorým daný používateľ nemá prístup a „Assigned“ predstavuje projekty daného používateľa.



Obr. č. 21 Rozhranie na pridávanie projektov používateľovi

Pre zvolenie jedného s používateľov musí administrátor svoju voľbu označiť a potvrdiť „Select“. Po zvolení používateľa sa zobrazí list projektov (priradených aj nepriradených) tomuto používateľovi. Pri získavaní projektov zvoleného používateľa sa používa metóda

```
public SwapperProjectModel getAllProjects(int userId)
        {
            var serializer = new JavaScriptSerializer();
            SwapperProjectModel model = new SwapperProjectModel();

            List<Project> userProjects =
DB_Methods.ProjectMethods.GetUserProjects(userId);
            List<Project> allProjects = DB_Methods.ProjectMethods.GetAllProjects();
            List<Project> noUserProjects = allProjects.Except(userProjects, new
ProjectComparer()).ToList<Project>();

            model.AssignedList = new List<SelectListItem>();
            foreach (var p in userProjects)
            {
                model.AssignedList.Add(new SelectListItem() { Text = p.Name, Selected
= true, Value = p.ProjectId.ToString() });
            }

            model.AvailableList = new List<SelectListItem>();
            foreach (var p in noUserProjects)
            {
                model.AvailableList.Add(new SelectListItem() { Text = p.Name, Selected
= true, Value = p.ProjectId.ToString() });
            }

            model.currentAvailableList = serializer.Serialize(model.AvailableList);
            model.currentAssignedList = serializer.Serialize(model.AssignedList);

            return model;
        }
```

Vstupom tejto metódy je id používateľa, ktorého projekty chceme získať. Výstupom je objekt, ktorý obsahuje informácie potrebné pre zobrazenie informácií administrátorovi.

Po zvolení „Add“ sa označený projekt z časti „Available“ priradí do listu priradených projektov a zároveň sa odstráni z listu nepriradených projektov. Po zvolení „Remove“ sa označený projekt v časti „Assigned“ zmaže z listu priradených projektov a priradí sa do listu nepriradených projektov.

Po zvolení „Save“ sú do databázy vkladané objekty z listu priradených projektov a naopak odstraňované projekty z listu nepriradených projektov, a to v metóde

```
public static void AddRemoveProjectsToUser(int u, List<int> addProjects, List<int>
removeProjects)
        {
            using (var dc = new CRDataContext())
            {
                foreach (var p in addProjects)
                {
                    if ((dc.UserProjects.Where(x => ((x.ProjectId.Equals(p)) &&
(x.UserId.Equals(u)))).Count()) == 0)
                    {
                        var project = new UserProject() { ProjectId = p, UserId = u };
                        dc.UserProjects.InsertOnSubmit(project);
                    }
                }
                foreach (var p in removeProjects)
                {
                    if ((dc.UserProjects.Where(x => ((x.ProjectId.Equals(p)) &&
(x.UserId.Equals(u)))).Count()) > 0)
                    {
                        var project = (from up in dc.UserProjects where up.ProjectId
== p && up.UserId == u select up).First();
                        dc.UserProjects.DeleteOnSubmit(project);
                    }
                }
                dc.SubmitChanges();
                return;
            }
        }
```

Testovanie

Výsledky boli testované manuálne porovnávaním s údajmi v databáze.

6.7 PRIRADENIE POUŽÍVATEĽOV K PROJEKTU

Tento príbeh nadväzuje na príbeh z prvého šprintu 4.4. Je dôležité, aby rozhranie bolo čo najjednoduchšie, a aby administrátor nemusel každého používateľa priradiť ku projektu individuálne. Je potrebné, aby bolo možné zvoliť skupinu používateľov, ktorí budú k projektu priradený.

Návrh

Je potrebné vytvoriť rozhranie, ktoré umožní multiselekciu používateľov.

Riešenie

Pri riešení bol zvolený rovnaký postup ako pri priradení projektov používateľovi (časť 6.6), s tým rozdielom, že projekty boli nahradené používateľmi a naopak.

Testovanie

Výsledky boli testované manuálne porovnávaním s údajmi v databáze.

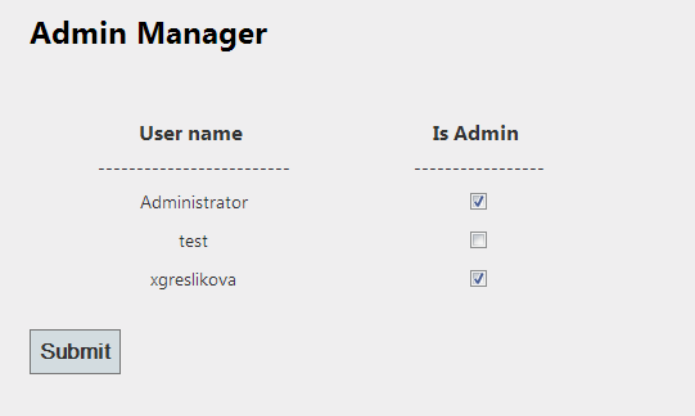
6.8 URČENIE ADMINISTRÁTOROV SYSTÉMU

Je nevyhnutné aby mal systém viac ako jedného administrátora. Tiež je potrebné aby časom bolo možné pridať nového administrátora systému no tiež odstrániť administrátorov, ktorí budú nečinní. Z tohto dôvodu je potrebné aby administrátori mali právo pridávať aj odstraňovať administátorov systému.

Analýza
Každý s používateľov má v systéme svoju rolu. Používateľ môže mať viacero rolí. Na pridávanie administrátorov má právo iba administrátor. Údaje o rolách jednotlivých používateľoch získame z databázy.

Návrh
V prvom kroku je potrebné získať z databázy všetkých používateľov a ich roly. Následne sa zistí, ktorí používatelia sú už administrátori systému. Tieto údaje sa zobrazia v prehľadnej tabuľke, kde bude prihlásenému administrátorovi umožnené zmeniť tieto údaje.

Riešenie
Na načítanie údajov o používateľoch a ich rolách v systéme bola použitá metóda `UserMethods.GetUsersAndRoles()`. Na základe týchto údajov bol vytvorený model pre každého používateľa , ktorý obsahoval základne údaje o používateľovi (Id a meno používateľa) a údaj či daný používateľ je, alebo nie je administrátorom systému. Tieto údaje boli zobrazené v tabuľke, ktorá obsahuje meno používateľa a checkbox. Ak checkbox je označený, znamená to, že daný používateľ je administrátorom systému. Náhľad sa nachádza na obrázku č. 22.



Obr. č. 22 Pridávanie administrátorov

6.9 DEFINOVANIE AKTÍVNYCH ODKAZOV V KONTEXTOVOM MENU

Úlohou bola pridávanie aktívnych odkazov do kontextové menu, ktorou sa mali zabezpečiť správne prepojenia medzi funkcionalitami systému.

Návrh

Aktívne odkazy pre entity:

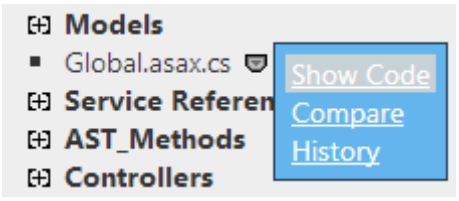
- zobrazenie kódu
- zobrazenie histórie
- porovnanie s predchádzajúcou verziou

Aktívne odkazy pre odovzdania:

- Porovnanie s prechádzajúcou verziou
- Zobrazenie detailov o odovzdaní

Riešenie

Do kontextové menu pri entitách boli pridané aktívne odkazy, ktoré boli navrhnuté. Aktívne odkazy boli pridané do View kontroléra *FileViewer*. Na obrázku č. 21 je znázornená menu s pridanými odkazmi.



Obr. č. 23 Kontextové menu pre entity s odkazmi

Aktívne odkazy pri entitách

- *Show Code* – po kliknutí sa zavolá príslušná metóda *FileRequest()* pre zobrazenie zdrojového kódu entity vedľa stromovej štruktúry projektu.
- *Compare* – zavolá sa metóda *FilesCompared()* kontroléra *EntityVersionController*, ktorá porovnáva zdrojový kód s predchádzajúcou verziou.
- *History* – zavolá sa metóda *EntityVersionController.Index()* pre zobrazenie histórie verzií danej entity

Pre zabezpečenie správnej funkcionality boli zmenené metódy kontroléra *EntityVersionController.*

Aktívne odkazy pri odovzdaniach

Pri zobrazeniach odovzdania existovali aktívne odkazy pre definované akcie. Tieto odkazy neboli v celku zmenené. Opravený bol iba nesprávne volanie zobrazenia zdrojového kódu v stromovej štruktúre zvýraznenými zmenami.

6.10 Úplné zobrazenie číslovania riadkov

Je potrebné upraviť číslovanie riadkov, ktoré je pri zobrazovaní obsahu súboru vždy po piatich.

Analýza
Problém treba riešiť drobnou úpravou kódu a zmenením triedy v HTML dokumente, ktorá je využívaná skriptom zabezpečujúcim formátovanie.

Návrh
Najvhodnejším riešením je úprava triedy "linenums".

Riešenie
Trieda "linenums:5" bola upravená na "linenums:1". Táto drobná úprava zabezpečí, že číslo riadku bude uvedené na každom riadku a nie na každom piatom riadku.

Testovanie
Testovanie prebehlo jednoduchým overením výpisu obsahu súborov. Overenie riešenia bolo úspešné.

6.11 Vizualizácia rozbalenia uzla stromu

Cieľom tohto príbehu je zabezpečiť jednoduchý vizualizačný prvok v rámci zobrazovaného stromu. Ak sa dá uzol stromu rozbaliť, pri jeho názve bude zobrazený príslušný piktogram a to isté platí aj ak sa uzol rozbaliť nedá.

Analýza
Je potrebné opäť zvážiť, akým spôsobom je možné docieliť interakciu v okne prehliadača pri ktorej sa priamo mení obsah zobrazovaného dokumentu. Najrozumnejšie riešenie spočíva v použití JavaScriptu, ktorý po kliknutí na uzol stromu zabezpečí zobrazenie vizuálneho príznaku pri uzle stromu. Keďže sme v minulom riešení použili knižnicu jQuery, aj v tomto prípade ju použijeme. Treba aj zvážiť, akým spôsobom docielime zobrazenie samotného piktogramu.

Návrh
Návrh riešenia spočíva v zachytávaní kliknutia v JavaScriptovom kóde. Následne zistíme, či bol uzol rozbalený alebo nie. Na základe tohto údaju priradíme príslušný piktogram dynamickou manipuláciou CSS dokumentu priamo v prehliadači.

Riešenie
Riešenie spočíva vo využití možnosti štýlovania neusporiadaného zoznamu. Základný zoznam využíva prednastavený piktogram plnej guličky pre jednotlivé položky zoznamu. Tento piktogram sa dá úplne odstrániť prípadne nahradiť obrázkom. Samotné riešenie teda spočíva v tom, že po kliknutí zmeníme štýl odrážky v zozname. To docielime definovaním si dvoch tried v našom CSS dokumente (obr. č. 22).

```
.plus
{
    list-style-image: url("../Images/li-plus.png");
}

.minus
{
    list-style-image: url("../Images/li-minus.png");
}
```

Obr. č. 24 Triedy v CSS dokumente

Tieto dve triedy definujú dva rôzne piktogramy. Následne už len stačí po kliknutí tieto dve triedy striedať (obr. č. 23).

```
$(".toggle").click(function () {
       $(this).toggleClass("plus");
       $(this).toggleClass("minus");
       $(this).next().slideToggle();
});
```

Obr. č. 25 Striedanie CSS tried

Testovanie

Testovanie bolo jednoduché. Stačilo overiť, či uzavretý uzol má vždy jeden typ piktogramu a otvorený uzol má vždy druhý typ piktogramu. Toto testovanie sa priamo spájalo s testovaním rozbaľovania stromu. Testovanie bolo úspešné.

6.12 PRÁZDNE RIADKY NA ZAČIATKU A KONCI KÓDU

Úlohou bola odstránenie nadbytočných znakov nového riadku, ktoré boli na začiatku a na konci zobrazených kódov.

Riešenie:

Tento problém vznikol kvôli bielych znakov medzi značkami *<pre></pre>* a reprezentáciou zdrojového kódu nejakej entity. Problém bol odstránení tým, že sa zmazali všetky biele znaky medzi značkami a spomenutou reprezentáciou

7 CELKOVÝ POHĽAD NA PROJEKT

Po troch šprintoch môžeme napísať priebežný pohľad na aktuálny stav projektu. Počas posledných týždňov pracovali všetci členovia tímu postupne na projekte a všetkých jeho častiach. Celková práca sa dá rozdeliť do dvoch základných kategórii.

7.1 TVORBA SOFTVÉROVÉHO PROJEKTU

Počas prvých troch šprintov bola vytvorená základná kostra projektu. Táto kostra je postavená na architektonickom vzore MVC a logickú štruktúru projektu rozdeľuje na 3 základné časti: Model, View a Controller. Boli zabezpečené základné funkcionality v projekte. Sú nimi registrácia a prihlásenie používateľa, vytvorenie administrátora, ktorý môže prideľovať projekty, zobrazovanie jednotlivých vlastností projektov. Boli taktiež zabezpečené funkcionality, ktoré nie sú priamo pre bežného používateľa viditeľné, ako napr. získavanie informácii o projektoch z verziovacieho systému AST-RCS, vytvorenie potrebných dátových štruktúr na reprezentáciu údajov alebo zabezpečenie zobrazovania informácii iba pre používateľa, ktorý má na to oprávnenie.

Okrem toho boli zabezpečené všetky ďalšie súčasti, ktoré boli nevyhnutné pre samotnú prácu na projekte. Bol nakonfigurovaný server, databázový systém ako aj systém na verziovanie.

7.2 TVORBA DOKUMENTÁCIE

Nevyhnutnou súčasťou našej práce bola aj priebežná tvorba dokumentácie, ktorá opisovala našu prácu na projekte v jednotlivých šprintoch. Okrem toho bol vytvorený ešte druhý dokument, v ktorom je rozoberané riadenie v rámci projektu. Tento projekt obsahuje zoznam členov tímu, opis ich manažérskych rolí, vypracované metodiky a ako prílohu obsahuje zápisnice z jednotlivých týždenných stretnutí. Spomínané metodiky boli vypracované do detailov a obsahujú presné opisy postupov pri rôznych činnostiach akými sú napr. tvorba dokumentácie alebo písanie zdrojového kódu.

8 VODIČKA

Číslo šprintu: 4

Začiatok šprintu: 20.11.2013

Koniec šprintu: 4.12.2013

Príbehy:

- **Zobrazenie zdrojového kódu skrýva dokumentačné tagy**
- **Chýbajúci obrázok kontextového menu v strome**
- **Pri zobrazení porovnania súboru bez histórie nastane chyba**
- **Písmo v zobrazení changesetov sa prekrýva**
- **Obnovenie sedenia pre údaje projektu**
- **Súbežné scrollovanie v porovnaní zdrojových kódov**
- **Zvýraznenie entity v zdrojovom kóde**
- **Zbalenie a rozbalenie celého stromu**
- **Infraštruktúra pre zaznamenávanie udalostí**
- **Oddelenie RCS projektov od logických projektov**
- **Správa logického projektu**
- **Správa požiadaviek o začlenenie RCS projektu k logickému projektu**
- **Správa používateľov logického projektu**
- **Správa aliasov používateľa**
- **Načítanie zoznamu značiek priradených prehliadanej súborovej entite**
- **Zobrazenie projektov - nastaviť pevnú šírku zoznamov**
- **Projekty sa administrátorovi načítajú až keď sa prihlási druhýkrát**

8.1 ZOBRAZENIE ZDROJOVÉHO KÓDU SKRÝVA DOKUMENTAČNÉ TAGY

Kvôli zistenej chybe (*Bug:360:* Zobrazenie zdrojového kódu skrýva dokumentačné tagy) bolo potrebné zistiť príčinu chyby.

Analýza

Problém pričinilo spôsob zobrazenia zdrojového kódu medzi značky *<pre>* a *</pre>*. Zobrazovaný obsah zdrojového súboru bol zabalený do metódy *Html.Raw()*, ktorý bránil zobrazeniu dokumentačných tagov.

Riešenie

Odstránením zabalenia obsahu do metódy *Html.Raw()*, sa podarilo zobrazovať súbor s celkovým obsahom.

8.2 CHÝBAJÚCI OBRÁZOK KONTEXTOVÉHO MENU V STROME

Je potrebné opraviť cestu k ikonke kontextového menu, ktorá sa nezobrazuje vo verzii programu bežiacej na našom serveri.

Riešenie

V tejto úlohe bolo riešenie veľmi priamočiare. HTML súbor, ktorý obsahoval cestu k ikonke, obsahoval nesprávnu cestu. Táto cesta fungovala v lokálne testovanej verzii projektu, ale na serveri už nefungovala. Samotné riešenie teda pozostávalo z jednoduchej úpravy cesty tak, aby sa ikonka zobrazovala korektne.

Testovanie

Testovanie prebiehalo overením funkčnosti na serveri. Testovanie bolo úspešné a ikonka sa zobrazovala korektne.

8.3 PRI ZOBRAZENÍ POROVNANIA SÚBORU BEZ HISTÓRIE NASTANE CHYBA

Súbor, ktorý bol novo vytvorený nemá žiadnu predchádzajúcu verziu takže ho nie je možné porovnať s inou verziou.

Riešenie

Používateľ po kliknutí na zobrazenie zmien v odovzdaní uvidí pri takýchto súboroch iba zdrojový kód nového súboru.

8.4 PÍSMO V ZOBRAZENÍ CHANGESETOV SA PREKRÝVA

Kvôli zistenej chybe (*Bug 348:*Písmo v zobrazení changesetov sa prekrýva) bolo nutné zisťovať príčinu a opraviť chybu, tak aby bolo zobrazenie grafu jasné a intuitívne pre používateľa.

Analýza

Zisťovania príčiny problému ukázalo, že nie je vhodné používať absolútne pozície opisov jednotlivých „changesetov“.

Ďalšie možnosti riešenia:

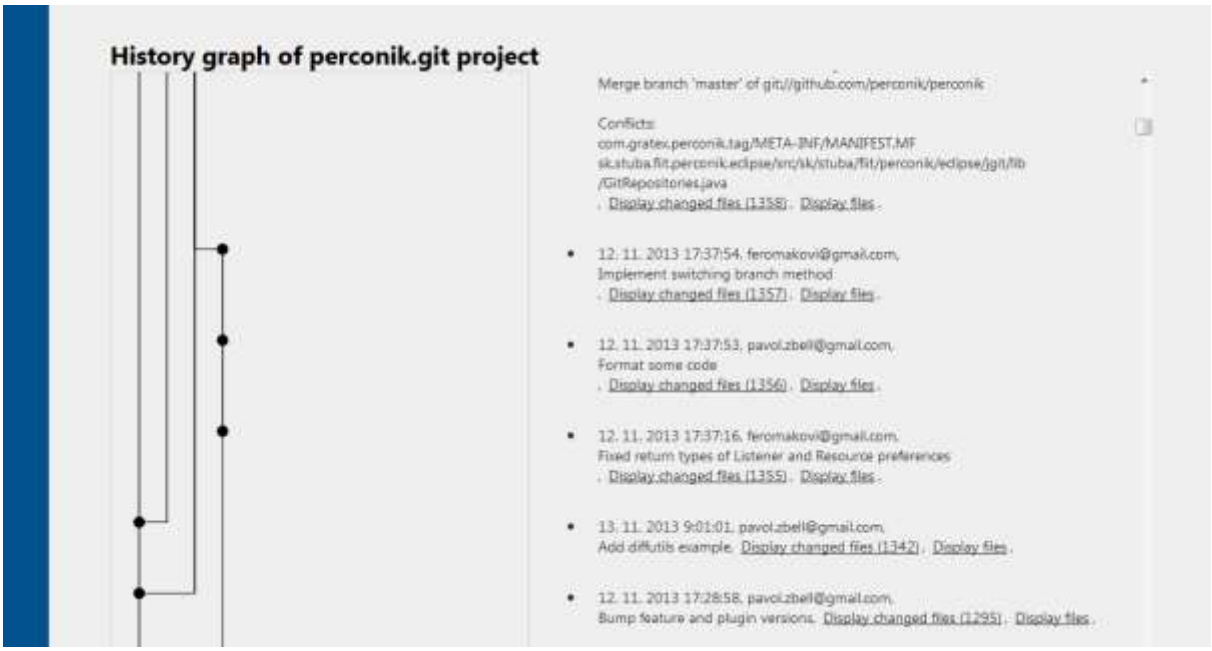
- Nastaviť pozície na relatívne a nechať prehliadač vyrenderovať tieto záznamy.
- Usporiadať záznamy do tabuľky.
-

Návrh
Lepšie riešenie z navrhnutých sa ukázalo využiť relatívne pozície pre záznamy. Keďže program bol už prispôsobený na spracovanie pozícií jednotlivých záznamov a priradenie ich k bodu, ktorý predstavuje „changeset“. Z tohto dôvodu bola táto zmena iba malá a nebolo nutné meniť zobrazovanie grafu.

Tiež bolo nutné ale prijateľné zmeniť poradenie záznamov, tak aby boli zoradené podľa času popisy „changesetov“ a nie body, ktoré ich predstavovali v grafe.

Riešenie
Pri úprave predchádzajúceho riešenia bolo nutné zmeniť respektíve odstrániť vypočítavanie pozícií pre záznamy, keďže ich pozícia bola zmenená na relatívnu.

Pre úpravu rozdelenia bodov v grafu bol použitý iný prístup vo vykresľovaní spojovacích čiar. Pre lepšiu viditeľnosť a prehľadnosť (Obr. č. 26).



Obr. č. 26 Upravený graf zobrazujúci históriu projektu.

Testovanie
Jednotkový test z predchádzajúceho riešenia zostal plne funkčný. Testovanie nového zobrazenia prebiehalo vizuálne na projekte „perkonik.git“, ktorý ma najzložitejšiu štruktúru vetiev z dostupných projektov.

8.5 Obnovenie sedenia pre údaje projektu

Po buildnutí programu sa Session uchovávajúca informácie o načítanom projekte zmaže.

Riešenie
Pre zjednodušenie práce boli údaje o session načítané do cookies a následne pri zmazaní session obnovené z cookies.

8.6 SÚBEŽNÉ SCOLLOVANIE V POROVNANÍ ZDROJOVÝCH KÓDOV

Porovnávanie dvoch verzií kódu sa zobrazuje v dvoch oddelených sekciách vedľa seba. Aby bola zabezpečená prehľadnosť pri porovnávaní zmien je potrebné synchronizovane scrollovať obidve sekcie kódu.

Riešenie

Implementovali sme JavaScript, ktorý vykonáva scrollovanie prvej sekcie podľa polohy druhej sekcie kódu.

8.7 ZVÝRAZNENIE ENTITY V ZDROJOVOM KÓDE

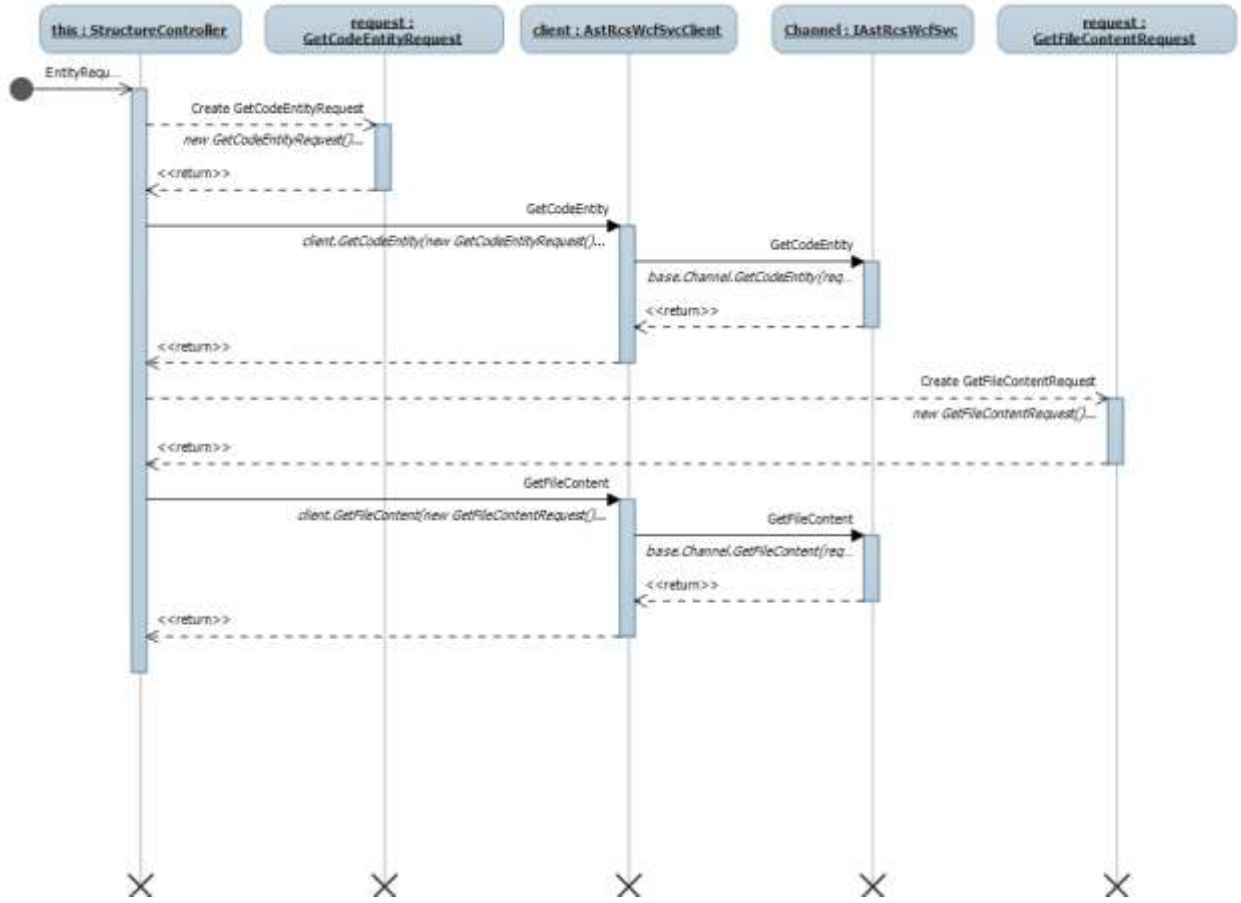
Používateľ chce mať zvýraznené miesto v zdrojovom kóde, ku ktorému je priradená kódová entita, aby vedel túto entitu rýchlo identifikovať bez nutnosti podrobnej analýzy kódu.

Po výbere kódovej entity, bude používateľovi zobrazený zdrojový kód, v ktorom je zvolená kódová entita. Po zobrazení kódu, bude nastavená pozícia v kóde na začiatok kódovej entity a obsah kódovej entity bude graficky zvýraznený.

Návrh

Táto funkcionalita systému je prístupná zo stromovej štruktúry entít. Po výbere *Structure->Entity Structure* používateľovi sa zobrazí spomenutá stromová štruktúra. Kliknutím na názov nejakej entity sa vykoná zvýraznenie entity. Zobrazenie požadovanej entity sa vykoná ako parciálne zobrazenie vedľa stromovej štruktúry. Samotná metóda na vykonanie spomenutej úlohy bude implementovaná do existujúceho kontroléra *StructureController*.

Riešenie



Obr. č. 27 Sekvenčný diagram metódy EntityRequest

Na obrázku č.27 je znázornená funkcionalita metódy *EntityRequest*. Metóda na základe *versionId* entity získa údaje o umiestnení entity v súbore, a zistí *fileVersionId*. Podľa *fileVersionId* získa obsah súboru, ktorý spolu informáciami o umiestnení entity v súbore slúži ako parameter pre parciálne zobrazenie. Zvýraznenie entity a rolovanie na jej začiatok sa rieši vo View, ktorý je na obrázku č. 28.

Obr. č. 28 Zvýraznenie entity vo View

Testovanie
Správna funkcionalita bola testovaná vizuálne a kontrolou údajov pomocou ladenia.

8.8 ZBALENIE A ROZBALENIE CELÉHO STROMU

Cieľom tohto príbehu je zabezpečiť jednoduchú funkcionalitu, pri ktorej bude možné stromovú štruktúru zbaliť resp. rozbaliť jedným kliknutím.

Analýza
Podobné funkcionality sme zabezpečovali už aj v minulosti a preto bude aj v tejto úlohe pravdepodobne použitý JavaScript v kombinácii s dobre známou knižnicou jQuery. Samotná funkcionalita, by potom mohla byť používateľovi zobrazená vo forme odkazu na ktorý musí kliknúť.

Návrh
Návrh riešenia bude pozostávať z dvoch častí:

- Úprava HTML - jednoduché pridanie dvoch odkazov do HTML dokumentu
- Vytvorenie JS kódu - pridanie jednoduchého JS kódu, ktorý zabezpečí danú funkcionalitu

Riešenie
Samotné riešenie bolo implementované nasledovne:

1. Do HTML dokumentu boli pridané dva odkazy. Každý odkaz má unikátne ID.
2. Bol vytvorený jednoduchý JS kód. Tento kód sa viaže na unikátne ID vytvorených odkazov. Po kliknutí na odkaz sa zavolá JS kód, ktorý využíva funkcionality jQuery knižnice na manipuláciu zobrazovania prvkov v dokumente.

Na obr. č. 29 môžeme vidieť ukážkový kód, ktorý sa stará o rozbalenie stromu.

```
$("#tree-expand").click(function () {
    $(".toggle").next().slideDown();

    return false;
});
```

Obr. č. 29 Kompletné rozbalenie stromu.

Testovanie
Testovanie prebehlo jednoduchým overením automatického rozbaľovania resp. zbaľovania stromovej štruktúry. Testovanie bolo úspešné.

8.9 INFRAŠTRUKTÚRA PRE ZAZNAMENÁVANIE UDALOSTÍ

Pre zaznamenávanie udalostí je nutné implementovať systém, ktorý bude spravovať súbor obsahujúci záznamy o udalostiach ako sú: chyby, informácie o stave alebo „debug“ výpisy.

Analýza
Analyzované boli dve možnosti logovania, ktoré sú najpoužívanejšie pre .NET aplikácie:

- NLog
- Log4Net

Návrh
Pre implementáciu logovania bol vybraný Log4Net pre jeho jednoduchosť v konfigurácií a implementácií.

Nový súbor sa generuje v momente keď starý dosiahne veľkosť 5mb, udržiavať sa bude 10 súborov. Ak počet súborov dosiahne 11 najstarší sa zmaže.

„Error“ metóda by mala obsahovať výnimku ako parameter ako aj správu o chybe a ako táto chyba nastala. Taktiež je nutné aby v zázname bola informácia o metóde kde táto chyba nastala a v kom čase.

„Info“ metóda by mala obsahovať správu ako parameter, ktorá opisuje stav/začatý proces v systéme.

„Debug“ metóda by mala obsahovať debug výpis ako parameter, tak aby pomohla odstrániť chyby.

Riešenie
Riešenie bolo implementované za pomoci dokumentácie knižnice Log4Net.
Zoznam „public“ metód:

```
public static void Error(string msg)

public static void Error(string msg, Exception ex)

public static void Error(Exception ex)

public static void Info(string msg)

public static void Debug(string msg)
```

Obr. č. 30 Public metódy poskytované triedou Logger.cs

Ukážka konfigurácie, ktorá zabezpečuje generovanie súboru za navrhnutých podmienok (Obr. č. 31).

```
<file value=".\Logs\ServerLog.txt" />
<appendToFile value="true" />
<rollingStyle value="Size" />
<maxSizeRollBackups value="10" />
<maximumFileSize value="5MB" />
```

Obr. č. 31 Časť konfiguračného súboru zabezpečujúci spôsob generovania a mazania súborov

Ukážka logu v súbore:

ERROR: 2013-12-04 12:47:22,824 >> MailClient.Send, Metóda: CodeReview.Web.Utils.MailClient.Send
System.Net.Mail.SmtpFailedRecipientException: Mailbox unavailable. The server response was: 5.7.1 <user@test.sk>...
Relaying denied. IP name lookup failed [147.175.160.146]
at System.Net.Mail.SmtpTransport.SendMail(MailAddress sender, MailAddressCollection recipients, String deliveryNotify,
Boolean allowUnicode, SmtpFailedRecipientException& exception)
at System.Net.Mail.SmtpClient.Send(MailMessage message)
at CodeReview.Web.Utils.MailClient.Send(String to, String subject, String body) in c:
\CodeReview\CodeReview.Web\Utils\MailClient.cs:line 25

Formát záznamu:

```
%level: %date >> %message%newline
```

Obr. č. 32 Formát záznamu udalosti

Použitie logovania je popísané v Metodike logovania, ktorá sa nachádza v dokumente Manažment riadenia.

Testovanie
Testovanie prebehlo na testovacích logoch, ktoré boli vložené do častí kódu. Boli testované všetky prekonania metódy Error(), aj s vložením výnimky (Výsledok testovania je v časti Riešenie – Ukážka logu v súbore).

8.10 ODDELENIE RCS PROJEKTOV OD LOGICKÝCH PROJEKTOV

Zadávateľ vyžaduje oddelenie RCS projektov od logických projektov, aby systém reálne kopíroval štruktúru projektov a používateľ mal možnosť spravovať prístup iných používateľov k projektu.

Analýza
Pre oddelenie RCS projektov od logických projektov bolo potrebné zmeniť databázový model, aby sme mohli ukladať RCS aj logické projekty. Logické projekty budú môcť obsahovať určitú množinu fyzických projektov.

Každý používateľ pričlenený k projektu má jednu z nasledovných rolí:

- Administrátor - môže priradiť používateľov k projektu a meniť im ich roly. Tvorca projektu je pri vytvorení nového projektu označený za administrátora.
- Editor - má právo editovať projekt (tzn. časom vytvárať a editovať značky)
- Čitateľ - má právo prehliadať projekt

Úloha pozostáva z nasledujúcich úloh:

- Zmena v štruktúre databázy
    - vytvorenie migračného skriptu
    - úprava dátového modelu
- Úprava atribútov pre nový model
- Úprava atribútov pre CR projekty
- Zobrazenie histórie odovzdaní projektu z vybraného CR projektu
- Pridanie výberu RCS projektu, s ktorým používateľ aktívne pracuje (tzn. prehliada zdrojové kódy a históriu)

Riešenie
Museli sme zmeniť dátový model. Pre lepšiu identifikáciu, o aký projekt sa jedná, sme zmenili názov tabuľky *Projects* na *CRProjects* (logický projekt). Tabuľku CRProjects sme doplnili o atribút *ProfileId*, ktorý identifikuje použitý profil značiek. Novú tabuľku sme nazvali *AstProjects* (fyzický projekt). Museli sme vytvoriť ďalšiu tabuľku, ktorá obsahuje roly používateľov pre projekt. Zmenu sme vykonali pomocou migrácii.

Obr.č.33 Rozšírený dátový model

UserProjectRoles – Tabuľka číselníkov, obsahujúca roly používateľov pre projekt (administrátor, editor, čitateľ)

- *UserProjectRoleId* – identifikačné číslo roly
- *Name* – názov projektovej roly

CRProject – tabuľka obsahujúca logické projekty

- CRProjectId – identifikačné číslo CR projektu
- Name – názov logického projektu
- ProfileId – identifikátor použitého profilu PerConik značiek

AstProject – tabuľka obsahujúca fyzické projekty (z AST-RCS)

- AstProjectId – identifikačné číslo AST projektu
- Name – názov projektu

AstInCrProjects – entitno relačná tabuľka, ktorá prepája tabuľky CRProjects a AstProjects

- Id – identifikačné číslo relačnej tabuľky
- AstProjectsId – referencia na fyzický projekt AstProjects
- CRProjectId – referencia na logický projekt CRProject

8.11 SPRÁVA LOGICKÉHO PROJEKTU

Do menu projektu je potrebné pridať položku na správu administrácie logického projektu.

Analýza

Do navigačného menu administrácie treba pridať novú položku tak, aby mal k nej prístup administrátor projektu.

Návrh

Pri zobrazovaní menu overíme, či je aktuálne prihlásený používateľ administrátor projektu a podľa toho zobrazíme príslušnú položku.

Riešenie

Pre overenie a zobrazenie sme implementovali nasledujúci kód:

```
@if (Session["User"] != null && Session["Project"] != null)
{
If(CodeReview.Web.Database.UserMethods.IsProjectAdministrator(((User)Session["User"]),
((ProjectProp)Session["Project"]).projectId))
        {
        <li><a href="#">Project administration</a>
              <ul>
                    <li style="width: 160px">@Html.ActionLink("Project users",
                     "SwapperUsers", "ProjectsManager")</li>
                    <li style="width: 160px">@Html.ActionLink("Request a repository",
                    "RequestRepository", "Project", new { projectId =
                    (((ProjectProp)Session["Project"]).projectId) }, null)</li>
              </ul>
        </li>
        }
}
```

Výsledná položka Project administration v menu na stránke je zobrazené na obrázku č. 34.

Obr. č. 34: Položka pre správu logického projektu

Testovanie

Funkcionalitu sme otestovali pomocou používateľa, ktorý vytvoril nový projekt a tým sa zároveň stal jeho administrátorom. Tomuto používateľovi sa zobrazila pridaná položka.

K vytvorenému projektu sme pridali druhého používateľa, ten však nebol jeho tvorca, čiže nemá administrátorské práva. Administrátorská položka v menu sa mu nezobrazila.

8.12 Správa požiadaviek o začlenenie RCS projektu k logickému projektu

Cieľom tohto príbehu je zabezpečiť funkcionalitu, kedy si používateľ v systéme môže k svojmu projektu požiadať o priradenie repozitára zo systému AST-RCS. Pokiaľ sa repozitár v systéme ešte nenachádza, používateľ má možnosť požiadať o priradenie projektu z iného verziovacieho systému pričom je požiadaný o zadanie špecifických informácií týkajúcich sa externého repozitára. Jeden projekt môže odkazovať na viacero repozitárov. Požiadavky sú spracované manuálne administrátormi systému, ktorí sú o nich upovedomení pomocou e-mailovej notifikácie. Následne admin môže požiadavku zamietnuť alebo povoliť.

Analýza

Tento príbeh sa skladá z troch hlavných častí:

1. Rozhranie pre používateľa + práca s databázou

2. Rozhranie pre administrátora + práca s databázou
3. E-Mailové notifikácie

Bude potrebné vykonať niekoľko úprav v databáze. Samotný systém bol koncipovaný od začiatku iným spôsobom. Je preto potrebné upraviť databázu tak, aby nám umožňovala priradiť viacero repozitárov k jednému projektu a aby jeden repozitár mohol byť priradený k viacerým projektom.

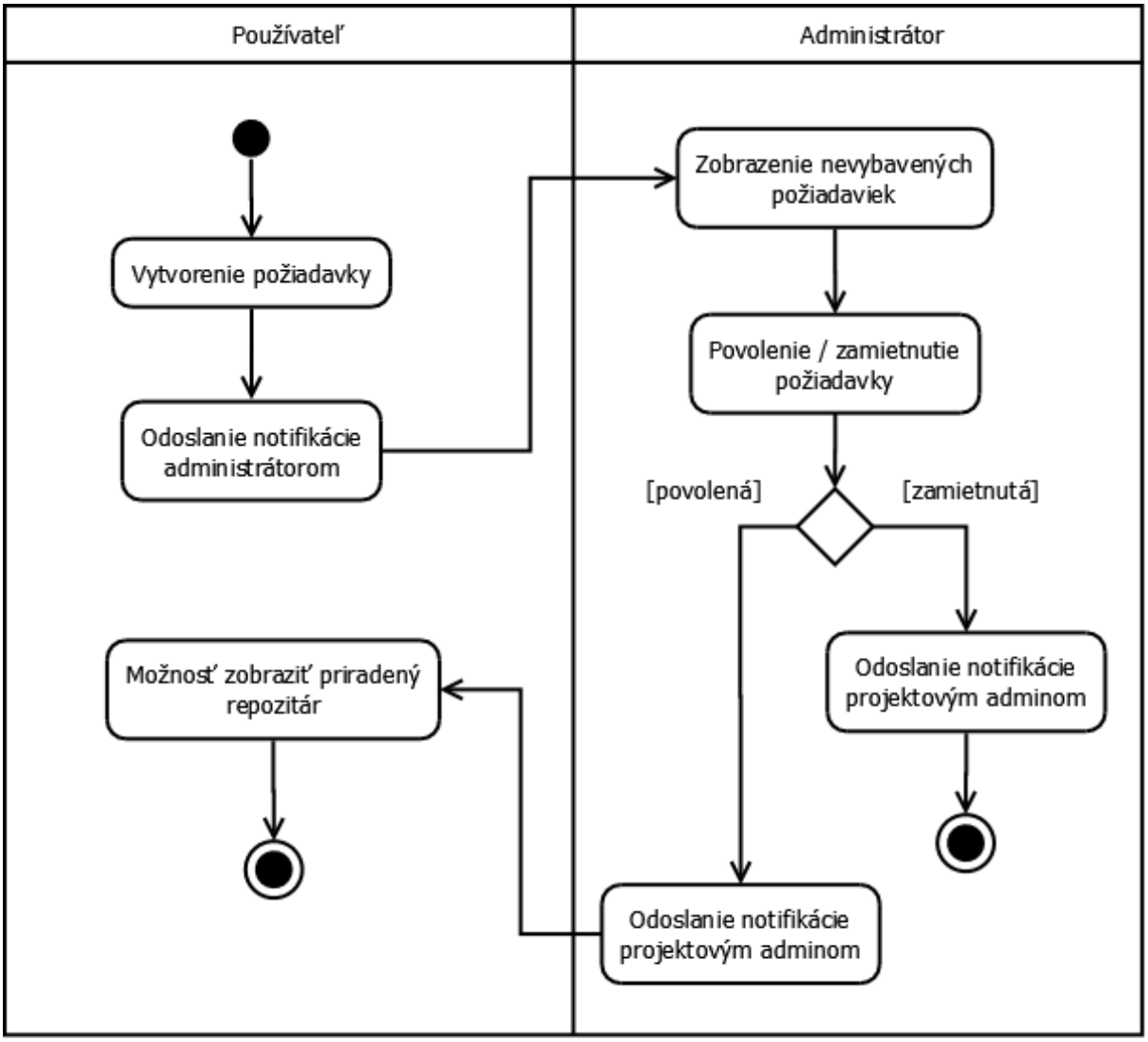
Pridávanie externých repozitárov do systému AST-RCS sa nachádza mimo nášho systému a preto tento problém nebudeme riešiť.

Návrh

Samotný návrh sa skladá z troch hlavných častí:

1. Rozhranie pre používateľa
    a. grafické rozhranie pre používateľa, formulár, ktorý používateľ vyplní a pošle administrátorom
    b. metódy na prácu s databázou s tabuľkou požiadaviek
2. Rozhranie pre administrátora:
    a. grafické rozhranie pre administrátora, tabuľka, kde sú zobrazené požiadavky
    b. pridať metódu na povolenie/zamietnutie požiadaviek
3. E-Mailový klient
    a. zvoliť vhodný SMTP server pre odchádzajúcu poštu
    b. vytvoriť pomocnú triedu na odosielanie e-mailov

Nasledujúci diagram aktivít znázorňuje, ako by mohla byť požiadavka prechádzať systémom (obr. 35).



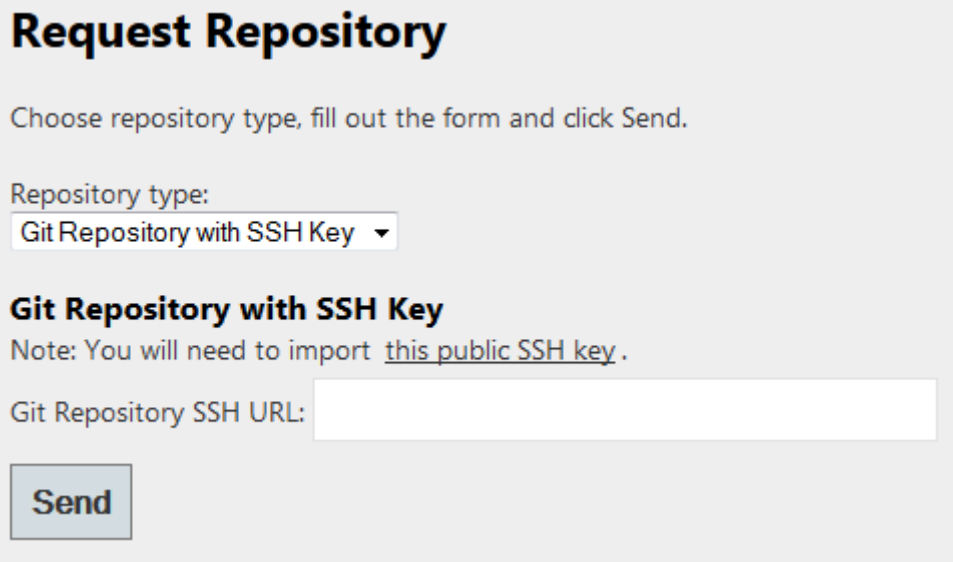
Obr. č. 35 Diagram aktivít pre správu požiadaviek.

Riešenie

Samotné riešenie je založené na návrhu.

1. Bola vytvorená metóda *RequestRepository* v triede *Project*. Táto metóda sa stará o formulár pre posielanie požiadaviek. K tejto metóde boli dopísané potrebné metódy na správu požiadaviek v databáze. Používateľ si môže zvoliť, ktorý typ repozitára chce priradiť a na základe typu sa mi zobrazí korektný formulár.
2. Bola vytvorená metóda *RequestManager* v triede *ProjectsManager*. Táto metóda sa stará o správu požiadaviek z pohľadu administrátora. Administrátor môže požiadavku povoliť alebo zamietnuť. Pri povolení musí napísať ID repozitára nachádzajúceho sa AST-RCS.
3. Bola vytvorená trieda *MailClient*. Táto trieda obsahuje metódu *Send*, ktorá umožňuje posielať e-mail. Ako SMTP server bol zvolený školský server.

Na obr. 36 sa nachádza ukážka formulára, ktorý vidí bežný používateľ. Formulár sa dynamicky mení na základe typu repozitára.



Obr. č. 36 Formulár pre odoslanie požiadavky.

Testovanie

Testovanie prebiehalo formou skúšania očakávanej funkcionality. Testovanie bolo úspešné. Neskôr bude pridaný jednotkový test, ktorý bude testovanie vykonávať automaticky.

8.13 SPRÁVA POUŽÍVATEĽOV LOGICKÉHO PROJEKTU

Administrátor projektu potrebuje stanoviť roly používateľom zainteresovaním na projekte.

Analýza

Vstupné podmienky:

- Používateľ je administrátorom projektu
- Administrátor projektu je prihlásený

Výstupné podmienky:

- Zmena práv používateľov zainteresovaných na projekte

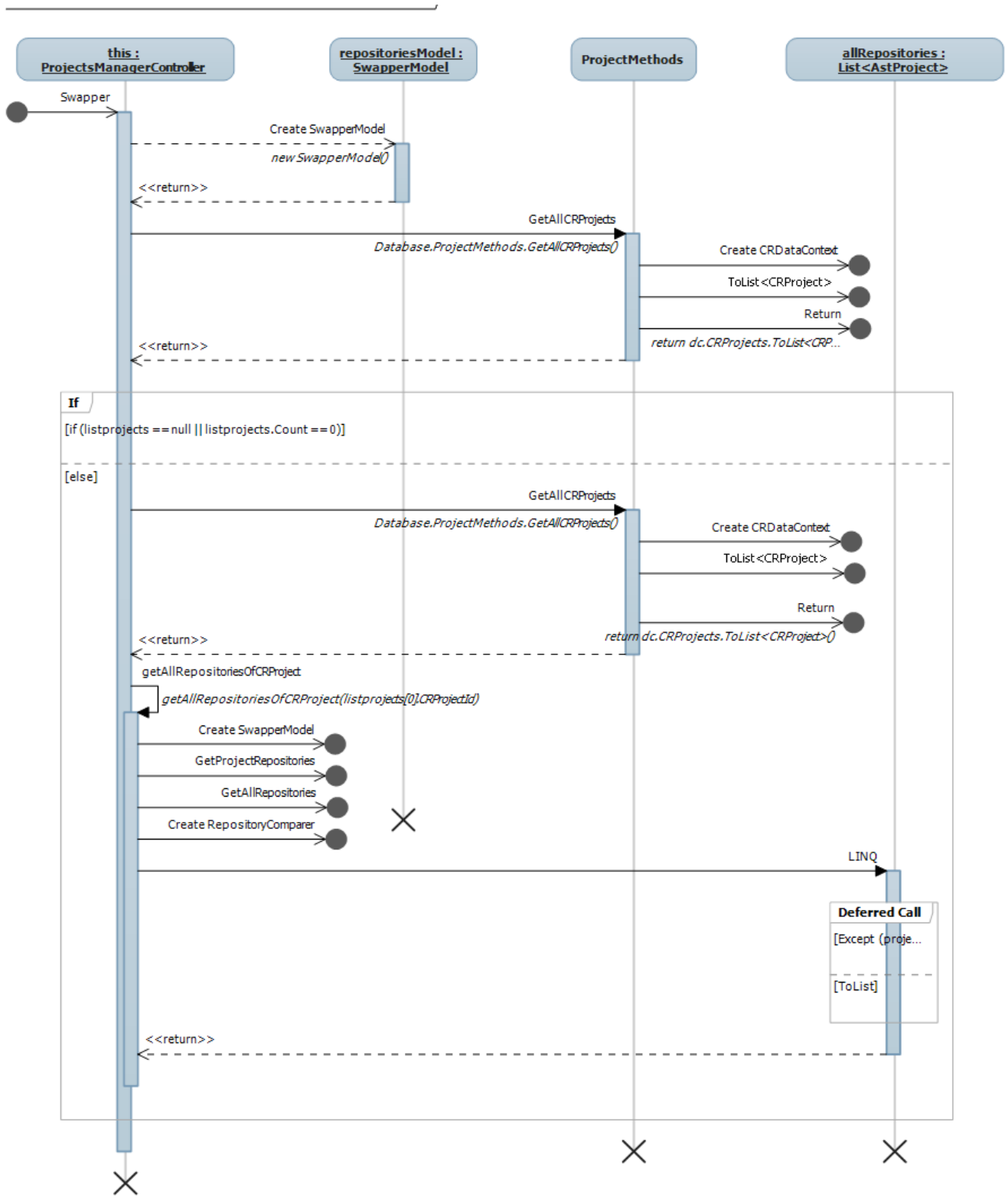
Účastníci: Administrátor projektu

Hlavný tok:

1. Administrátor projektu zvolí položku správa používateľov projektu.
2. Systém zobrazí všetkých používateľov, ktorý sú k danému projektu priradený.
3. Administrátor projektu zvolí roly pre jednotlivých používateľov alebo zmení súčasné roly používateľov .
4. Administrátor potvrdí výber.
5. Systém uloží zadané zmeny.
6. Prípad použitia končí.

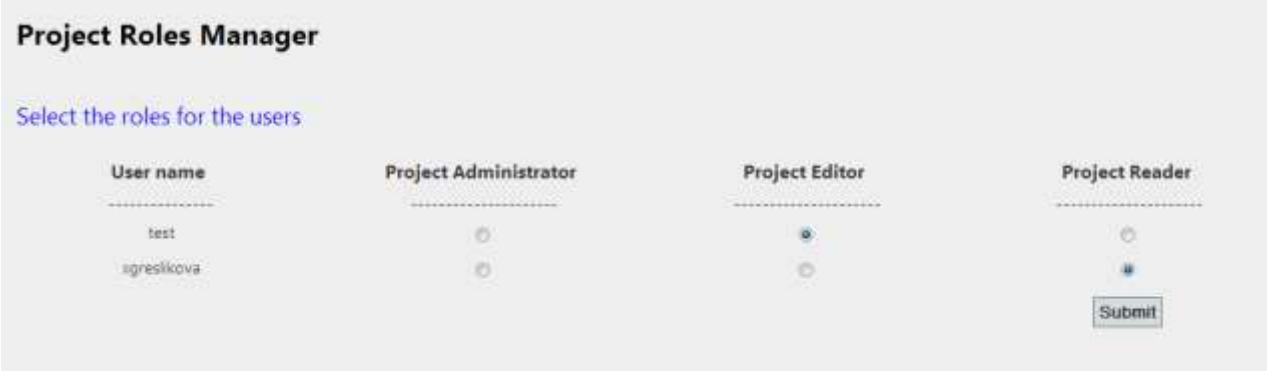
Návrh

Sekvenčný diagram sa nachádza na obrázku č. 37



Obr. č. 37 Sekvenčný diagram pre vytvorenie modelu

Administrátorovi projektu sú jednotlivý používatelia priradený k projektu zobrazený v prehľadnej tabuľke, ako na obrázku č. 38. Používateľ si zvolí roly pre jednotlivých používateľov. Po zvolení submit je jeho výber uložený.



Obr. č. 38 Zobrazenie správy používateľov

Riešenie

Čísla changesetov: 687, 648 , 647

Controller:

CodeReview.Web.Controllers.UserControllers

metódy:

[HttpGet]
public ActionResult ProjectRolesManager(int projectId)
[HttpPost]
public ActionResult ProjectRolesManager(IList<UserRoleInProject> userList)

Model:

CodeReview.Web.Models.UsersRolesModel.UserRoleInProject

View:

CodeReview.Web.Views.Users. ProjectRolesManager
CodeReview.Web.Views.Users.EditorTemplete.UserRoleInProject

Testovanie

Test1:

Vstup: Zvolenie projektu, ktorý nemá stanoveného žiadneho používateľa.

Výstup: Zobrazenie prebehlo úspešne. Zobrazila sa hláška „You don't have any users in the current project! Contact administrator.“

Test2:

Vstup: Korektné zvolenie rolí používateľov. Zvolenie jedného editora projektu a jedného čitateľa projektu. Uloženie.

Výstup: Prípad prebehol korektne. Zadané zmeny boli uložené do databázy. Následne sa administrátorovi projektu zobrazila potvrdzujúca informácia.

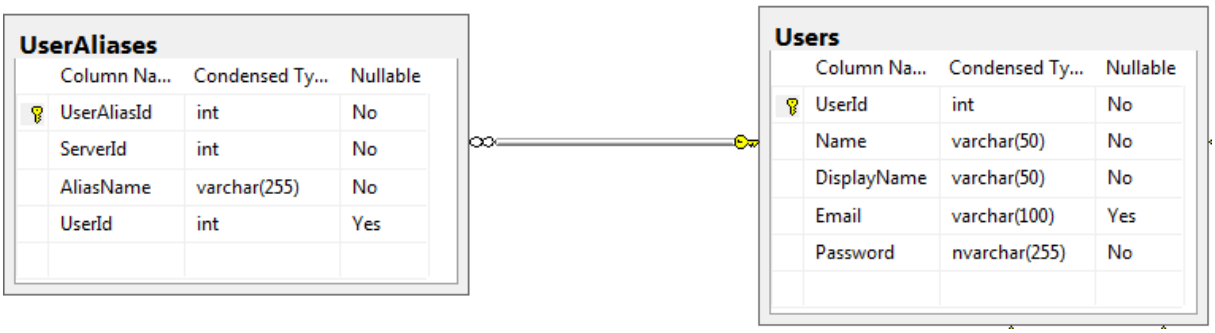
8.14 SPRÁVA ALIASOV POUŽÍVATEĽA

Používateľ chce mať možnosť nastaviť svoje aliasy na jednotlivých serveroch, aby mohol byť vo vytváranom systéme jednoznačne identifikovaný pod jedným kontom.

Analýza

Pre pridávanie aliasov je potrebné pridať novú tabuľku do databázy. Tabuľka sa má používať na ukladanie používateľských aliasov. Každý používateľ môže mať viac aliasov na jednom serveri. Aliasy sú unikátne, čo znamená že k rovnakému serveru nemôžu byť nastavené dve rovnaké aliasy.

Riešenie



Obr. č. 19 Tabuľka aliasov používateľov

Do databázy bola pridaná tabuľka ktorá je na obrázku č. 39 a jej podrobnosti sú nasledujúce

- *UserAliasId* – identifikačné číslo aliasu
- *ServerId* – identifikačné číslo servera, ku ktorému je alias priradený
- *AliasName* – meno aliasu
- *UserId* – referencia na používateľa, ktorému alias patrí

V rámci šprintu Vodička boli implementované nasledujúce databázové metódy na prácu s prvkami tabuľky *UserAliases:*

- `bool InsertAlias(int UserId, int ServerId, string AliasName)` – metóda na ukladanie aliasov do databázy, testuje unikátnosť, neexistuje dvojica *ServerId – AliasName* pridá nový záznam a vráti *true*, v inom prípade vráti *false*
- `bool Check(int ServerId, string AliasName)` – pomocná funkcia pre testovanie unikátnosti dvojice *ServerId a AliasName* v tabuľke
- `DeleteAlias(int UserAliasId)` – metóda na vymazanie záznamu z tabuľky podľa unikátneho klúča *UserAliasId*
- `GetAliasesOfUser(int UserId)` – funkcia vráti všetky záznamy používateľa, ktorý je identifikovaný podľa UserId

V šprinte Vodička príbeh nebol dokončený, neboli implementované časti požívateľského rozhranie. Chýbajúce sa časti budú dokončené v ďalšom šprinte.

8.15 Načítanie zoznamu značiek priradených prehliadanej súborovej entite

Pri práci je možné do zdrojového kódu súboru vpisovať rôzne značky, slúžiace pre iných členov tímu. Úlohou tohto príbehu je zobrazenie značiek pri zobrazovaní zdrojového kódu súboru na našej webovej stránke.

Analýza

Pridávanie značiek do zdrojového kódu je súčasťou Perconik nástrojov. Tieto značky pridávame pomocou pluginu vo vývojovom prostredí, v našom prípade Microsoft Visual Stuido 2012.

Našou úlohou je však značky získať a spracovať. Na to použijeme knižnicu *ITMaintanence.Driver.dll*, ktorá slúži po pripojení na server napríklad na vytiahnutie značiek z konkrétneho súboru, ktoré následne spracujeme a zobrazíme.

Značky budeme zobrazovať pri zdrojovom kóde súboru na našej stránke. V návrhu opíšeme spôsob zobrazenia a vyznačenia jednotlivých značiek.

Návrh
Značky budeme získavať zo zdrojového kódu súboru pomocou analyzovanej knižnice. Vytvoríme vlastný model, ktorý bude obsahovať popis značky, ktorým je typ, telo, začiatočný a koncový riadok v zdrojovom kóde súboru. Po pripojení na server zadáme požiadavku na získanie značiek pomocou ID verzie súboru. Zo všetkých značiek vyfiltrujeme požadované značky, konkrétne *UserInformationTag.* Tie zaznamenáme do listu a ten použijeme ako model, ktorý pošleme na zobrazenie.
Zobrazenie značiek vykonáme vedľa a v zdrojovom kóde súboru na stránke. Vedľa kódu vytvoríme pásik, ktorý bude obsahovať vyznačenie značky na správnom riadku, aby sa zaistila prehľadnosť voči zdrojovému kódu. Po prejdený nad toto označenie sa nám orámuje príslušný blok zdrojového kódu, zodpovedajúci značke. Ak na označenie klikneme, zobrazí sa vo vyskakovacom okne jej popis.

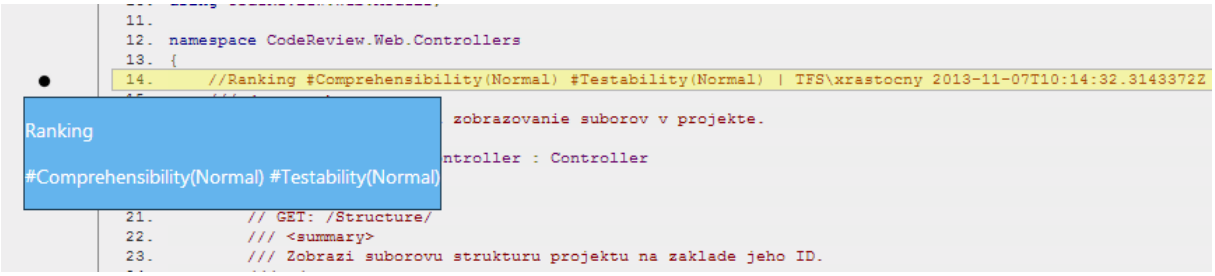
Riešenie
Prvou úlohou bolo pridanie knižnice *ITMaintanence.Driver.dll* do projektu. Po jej pridaní sme implementovali funkciu, ktorá získala list značiek požadovaného súboru. Tú sme naprogramovali v súbore *LoadTags.cs.*
Postup získania značiek je taký ako sme popísali v návrhu. Jeho implementácia teda pozostáva z nasledujúcich častí:

- Pripojenie k serveru
- Získanie značiek podľa ID verzie súboru
- Filtrácia značiek a ich zaznamenanie do modelu

Model sme následne poslali na zobrazenie. Pri zobrazení bolo potrebné pridať pásik s vyznačením umiestnenia značiek vedľa zobrazeného zdrojového kódu. Ten sme umiestnili naľavo od zdrojového kódu. Na ňom zobrazujeme bodky, ktoré majú tak definovanú pozíciu, aby sa zobrazili presne vedľa riadku alebo bloku v ktorom sa nachádza značka. Ďalšou pridanou funkciou je orámovací blok v texte zdrojového kódu, ktorý presne vyznačí umiestnenie značky a jej príslušného bloku v kóde. Posledným prvkom je pridané vyskakovacie okienko obsahujúce informácie o značke.

Na to, aby jednotlivé súčasti mohli spolu fungovať a dynamicky vykonávať potrebné úlohy tákajúce sa značiek, sme vytvorili JavaScript súbor *tags.js*, ktorý obsahuje funkcie napísané pomocou *jQuery*. Tieto funkcie zabezpečujú správne umiestnenie značiek, umiestnenie orámovania v kóde a zobrazenie vyskakovacieho okna. Funkcie zachytávajú správanie myši a podľa toho vykonávajú zodpovedajúce navrhnuté funkcie pri práci s bodom vyznačujúcim značku.

Výsledok môžeme vidieť na obrázku č.40.



Obr. č.40: Funkcionalita po prejdení a kliknutí na bod označujúci značku v kóde. Na obrázku môžeme pozorovať žlté orámovanie značky a po kliknutí aj zobrazené modré vyskakovacie okno s popisom značky.

Testovanie

Pri testovaní sme pozorovali správnosť umiestnenia nových prvkov pri zobrazení zdrojového kódu. Pri použití internetových prehliadačov Google Chrome a Mozilla Firefox sa bodky označujúce značku, orámovanie a vyskakovacie okno zobrazili presne na potrebný riadok alebo riadky v prípade párovej značky.

Problém však nastal pri prehliadači Internet Explorer. Riadkovanie zdrojového kóde tam má inú veľkosť a preto absolútna pozícia vyznačenia značky nie je pri správnom riadku zdrojového kódu. S touto problematikou sa treba v budúcnosti zaoberať a danú chybu vyriešiť.

8.16 Zobrazenie projektov - nastaviť pevnú šírku zoznamov

Pri zobrazení projektov nebola nastavená pevná šírka zoznamov, čo spôsobilo v prípadoch, keď sa v zozname nenachádzala žiadna položka deformáciu (zúženie) zoznamu.

Návrh

Potrebné nastaviť zoznamom pevnú šírku.

Riešenie

Zoznamu bola nastavená pevná šírka. `@style = "width: 200px;`

Keďže tento problém bol aj pri zobrazení používateľov zmeny boli vykonané vo „views“ :

Views\ProjectsManager\Swapper

Views\ProjectsManager\SwapperUsers

Testovanie

Zobrazenie zoznamov ktoré neobsahovali žiadne prvky. Zoznam nebol zdeformovaný (zúžený).

8.17 Projekty sa administrátorovi načítajú až keď sa prihlási druhýkrát

Pri načítavaní projektov dochádzalo k chybe pri ich načítavaní. Respektíve načítanie projektov bolo umiestnené v nesprávnej metóde.

Riešenie

Presunutie metódy `ProjectMethods.ReadProjectsFromASTRCS()` do „controller-a“ Controllers\AccountController.cs do metódy `Login(User user)`.

9 SLNIEČKO

Číslo šprintu: 5

Začiatok šprintu: 4.11.2013

Koniec šprintu: 11.12.2013

Príbehy:

- **Kontrola projektových práv**
- **Zmazanie starých projektov**
- **Vytvorenie a naštýlovanie systémového administračného menu**
- **Výška riadkov pri zvýrazňovaní kódu nie je určená správne**
- **Doplnenie informácií o požiadavke na začlenenie repozitára do projektu**
- **Zobrazenie kódu po kliknutí na meno súborovej alebo kódovej entity v strome**
- **Zvýraznenie entity v kóde žltým obdĺžnikom**
- **Zlúčenie menných priestorov v AST strome**
- **Zobrazenie tabuľky histórie entity vedľa stromu**
- **Premenovanie položiek Compare, History a Graph v kontextovom menu v strome**
- **Kontextové menu v AST strome a histórii**
- **Výber iba dvoch verzií entít na porovnanie**
- **Zachovanie hlavného pohľadu po zmene repozitára**
- **Systémová správa projektov a repozitárov**
- **Štatistiky značiek sú vždy počítané pre repozitár WhereIsMyCode**
- **Správa používateľov logického projektu**
- **Správa aliasov používateľa**
- **Základné štatistiky nad projektom**
- **User s administrátorskými právami si môže zrušiť administrátorské práva**
- **Stránkovanie pri načítaní AST projektov do databázy**

9.1 KONTROLA PROJEKTOVÝCH PRÁV

Používateľ požaduje, aby bol zabezpečený prístup k projektovým dátam aby sa nepoverené osoby nedostali k citlivým údajom.

Analýza

Vytvorením novej štruktúry projektov sa zmenila kontrola prístupu k dátam. Z toho dôvodu vznikli bezpečnostné diery, ktoré dovolili používateľovi prezerať jemu nepridelené projekty. Po zmene adresy v prehliadači na domovskej stránke projektu bolo možné zobraziť projekt, ktorý nie je používateľovi pridelený.

Riešenie

Riešením bolo vytvorenie nového atribútu *[AccessDeniedHomeProject]* , ktorý kontroluje parametre v adrese. Vytvorenie tohto atribútu bolo nutné preto, lebo pri vybratí logického projektu nie je vybraný žiaden *repozitár* (fyzický projekt).

Testovanie

Pri testovaní sme overovali, či sa používatelia po zmene adresy v prehliadači dokážu dostať na obsah, ktorý podľa systému nemajú právo prezerať. Pri všetkých testovaných prípadoch atribút *AccessDeniedHomeProject* zablokuje prístup k takémuto obsahu.

9.2 ZMAZANIE STARÝCH PROJEKTOV

Zákazník vyžaduje zmazanie starých projektov platných pred vytvorením logických projektov pre zvýšenie prehľadnosti databázy.

Analýza

Vytvorenie novej štruktúry projektov zmenilo prístup k dátam. V starej verzii používateľ mohol pracovať priamo s AST projektmi, bez spojenia s ostatnými projektmi. Zmena štruktúry dátového modelu ovplyvnila celý systém práce s projektmi, preto boli dočasne AST projekty uložené v tabuľke *CRProjects* dovtedy, dokým nebol celý systém upravený na prácu s novým dátovým modelom.

Po prechode na nové rozdelenie boli AST projekty uložené ako AST projekty ale aj ako CR projekty z dôvodu testovania a možnosti práce ostatných členov tímu. Aby zmena štruktúry bola úplná, museli sme vymazať AST projekty z tabuľky *CRProjects,* pretože v novej štruktúre CRProject je projekt, ktorý môže obsahovať niekoľko AstProjektov, čiže repozitárov.

Riešenie

Riešením bolo vytvorenie migračného skriptu, ktorý vymazal dáta z tabuľky *CRProjects* a z dôvodu prepojenia s ostatnými tabuľkami boli dáta vymazané z nasledujúcich tabuliek:

- *UserProjects*
- *Requests*
- *AstInCrProjects*
- *CRProjects*
- *AstProjects*

Všetky tieto tabuľky obsahovali dáta, ktoré boli prepojené s tabuľkou *CRProjects*. Po vymazaní dát sa spustila inicializácia AST projektov, ktorá načítala dáta už len do tabuľky *AstProjects*.

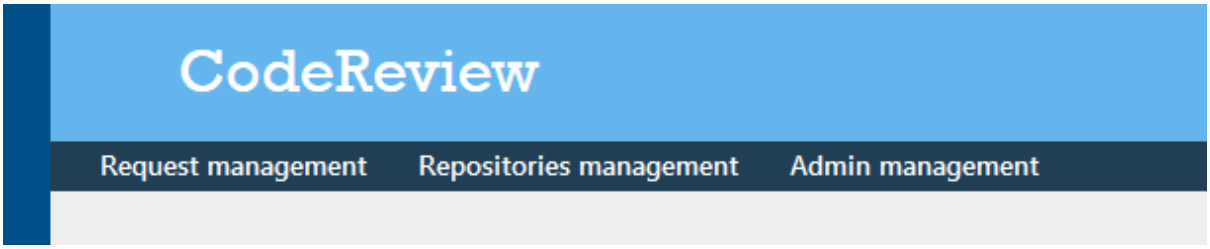
9.3 VYTVORENIE A NAŠTÝLOVANIE SYSTÉMOVÉHO ADMINISTRAČNÉHO MENU

Používateľ vyžaduje jednotné administračné menu systému aby mohol systém pohodlne spravovať.

Analýza
Menu bude vytvorené rovnakým spôsobom ako menu projektu. Položky administračného menu sú nasledovné:

- Administrácia požiadaviek
- Administrácia repozitárov
- Správa administrátorov

Riešenie

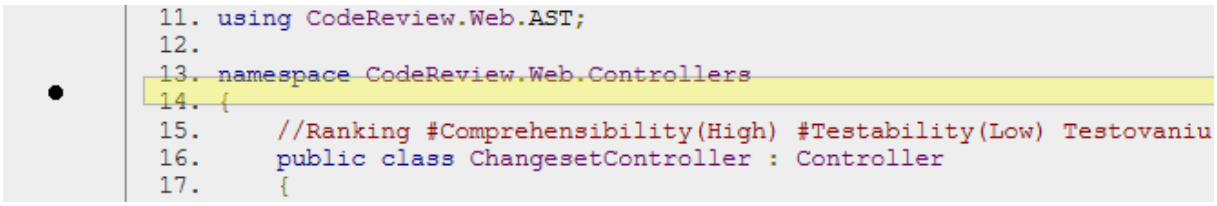


Obr. č. 41 Menu systémovej administrácie

9.4 Výška riadkov pri zvýrazňovaní kódu nie je určená správne

V šprinte Slniečko sme vypracovali User story s názvom Načítanie zoznamu značiek priradených prehliadanej súborovej entite. Pri testovaní výsledku nastal problém pri zobrazení značiek v prehliadači Internet Explorer.

Analýza
Pri testovaní v prehliadači Internet Explorer nastal prípad, keď umiestnenie značky nezodpovedalo riadku v kóde. Chybu si môžeme všimnúť na obrázku č.42.



Obr. č. 42: Nesprávne vyznačenie značky v internetovom prehliadači Internet Explorer

Problém nastal kvôli nesprávnemu určeniu výšky riadku zobrazeného zdrojového kódu. V našej implementácií je výška riadku nastavená na pevnú hodnotu 14 a pozícia zobrazených objektov je vypočítaná pomocou čísla počiatočného a koncového riadku značky, ktoré sú uložené v jej vlastnostiach.

Návrh
Pri riešení je potrebné poznať výšku riadku kódu. Druhou možnosťou je určiť presné umiestnenie riadku počas jeho výpisu. My si zvolíme pre implementáciu druhú možnosť.

Riešenie
Výpis doterajšieho riešenia sme značne upravili. To bolo riešené tak, že sa vypísal celý kód a osobitne sa potom pridávali grafické prvky. V terajšom riešení sme najprv usporiadali zoznam značiek podľa výskytu v zdrojovom kóde. Následne sme urobili výpis zdrojového kódu postupne po riadkoch, pričom sme si zapamätali umiestnenie riadkov výpisu, kde sa začína a kde končí značka. Na základe zapamätaného umiestnenia tak vieme vhodne umiestniť grafické prvky presne podľa riadkov. Výsledok úpravy pre porovnanie voči chybovému stavu môžeme pozorovať na obrázku č.43.

```
11. using CodeReview.Web.AST;
12.
13. namespace CodeReview.Web.Controllers
14. {
15.     //Ranking #Comprehensibility(High) #Testability(Low) Testovaniu
16.     public class ChangesetController : Controller
17.     {
```

Obr. č.43: Opravené orámovanie a umiestnenie pomocnej značky v internetovom prehliadači Internet Explorer

Testovanie

Testovanie prebiehalo po vizuálnej stránke. Kontrolovali sme správne umiestnenie prvkov a označenie všetkých značiek v prehliadanom zdrojovom kóde. Pôvodné nedostatky v prehliadači Internet Explorer boli odstránené. V testovaných prehliadačoch Internet Explorer a Mozilla Firefox nedochádzalo ku chýbam v zobrazeniach jednotlivých prvkov.

9.5 DOPLNENIE INFORMÁCIÍ O POŽIADAVKE NA ZAČLENENIE REPOZITÁRA DO PROJEKTU

Cieľom tohto príbehu bolo doplnenie niektorých detailov do e-mailových správ týkajúcich sa priraďovania repozitárov. Okrem toho bol upravený spôsob, akým je zobrazovaný popis požiadavky administrátorovi.

Analýza

Tento problém je riešiteľný drobnými úpravami v tele e-mailovej správy. Okrem toho bude nutné upraviť spôsob, akým popis požiadavky ukladáme do databázy. Následne už bude jednoduché zobrazovať popis požiadavky elegantnejším spôsobom.

Návrh

Ako prvé by mali byť doplnené e-mailové správy o ďalšie relevantné údaje. Následne sa upraví spôsob zobrazovania popisu požiadavky.

Riešenie

Samotné riešenie bolo implementované tak, ako bolo navrhnuté. Najprv sa doplnili údaje do mailovej správy. Bol pridaný priamy odkaz pre administrátorov, aby mohli rýchlo pristupovať do systému požiadaviek. Okrem toho bol zmenený spôsob ukladania popisu. Ten teraz používa HTML značku "
". Na samotné zobrazenie popisu je potom použitá pomocná funkcia na výpis HTML kódu (obr. 44).

```
<td>@Html.Raw(request.Description)</td>
```

Obr. č. 44 Priamy výpis HTML kódu.

Testovanie

Testovanie sa nezmenilo oproti pôvodnej implementácií, preto bolo vykonané len systematicky kontrolou obsahu e-mailovej správy a výpisu popisu požiadavky.

9.6 ZOBRAZENIE KÓDU PO KLIKNUTÍ NA MENO SÚBOROVEJ ALEBO KÓDOVEJ ENTITY V STROME

Cieľom tohto príbehu bolo obnovenie staršej funkcionality, ku ktorej sme sa vrátili.

Riešenie

Riešenie bolo priamočiare. Odkaz na zobrazenie súboru bol presunutý priamo pod názov súboru. Tento odkaz bol z kontextového menu odstránený.

Testovanie
Testovanie bolo vykonané skontrolovaním funkčnosti daného odkazu v HTML dokumente.

9.7 ZVÝRAZNENIE ENTITY V KÓDE ŽLTÝM OBDĹŽNIKOM

Používateľ požaduje zvýrazňovať entity v kóde žltým rámikom aby bola zachovaná konzistencia voči zobrazovaniu značiek.

Analýza
V rámci predchádzajúceho šprintu bolo pridané zvýraznenie značiek v zdrojových kódoch. Na takéto zvýraznenie bol použitý žltý rámik. Kvôli dodržiavania konzistentného vzhľadu bolo rozhodnuté zvýraznenie kódových entít v zdrojových súboroch rovnakým spôsobom.

Riešenie
Na zvýraznenie sa použilo v analýze definovaný rámik. Zmeny boli implementované v existujúcom súbore *Views/Structure/_EntityRequest.cshtm.*

9.8 ZLÚČENIE MENNÝCH PRIESTOROV V AST STROME

Zobrazovanie štruktúry kódových entít bolo duplicitné z dôvodu duplicity identifikátorov entít v systéme AST-RCS. Preto bolo nutné vymazať duplicity pre jasné zobrazovanie.

Riešenie
Riešenie spočíva vo filtrovaní duplicít za pomoci názvov entít. Tieto názvy boli rovnaké v niekoľkých entitách, avšak identifikátor bol rozdielny. Entity, ktoré majú rovnaké názvy boli identifikované a následne bol zvolený reprezentant, ktorý bude zobrazený . Identifikátor rodiča, ktorý sa rovnal zrušenej entite bol zmenený na identifikátor entity reprezentanta daného názvu.

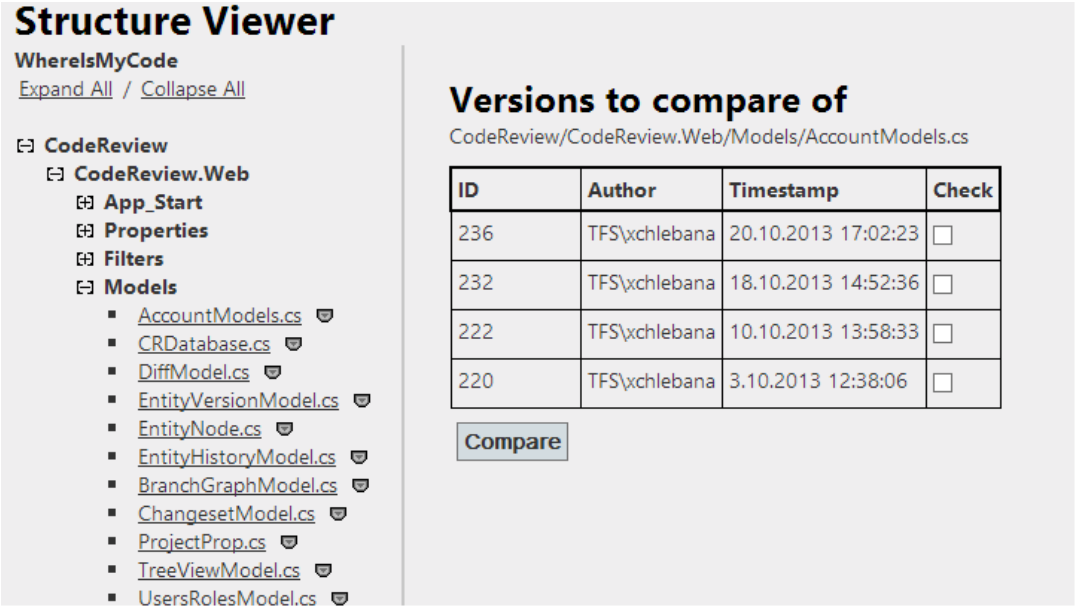
Testovanie
Testovanie prebehlo vizuálne a kontrolou údajov pri „debugu“ pri projekte s identifikátorom 3 (CodeReview).

9.9 ZOBRAZENIE TABUĽKY HISTÓRIE ENTITY VEDĽA STROMU

Používateľ vyžaduje zobrazenie tabuľky s históriou entity vedľa stromu, aby bola zachovaná konzistentnosť navigácie.

Analýza
Po kliknutí na zobrazenie tabuľky s históriu entity, spomenutá tabuľka sa zobrazí v novom View. Pre zobrazenie vedľa stromovej štruktúry je potrebné vytvoriť parciálne zobrazenie tabuľky.

Riešenie
Zobrazenie tabuľky s históriou entity bola prerobená na parciálne zobrazenie(Obrázok č. 45).



Obr. č. 25 Tabuľka s históriou entity vedľa stromovej štruktúry

9.10 Premenovanie položiek Compare, History a Graph v kontextovom menu v strome

Zákazník požaduje zmenu mena položiek Compare, History a Graph v kontextovom menu stromu aby bolo ovládanie intuitívne aj pre nových používateľov systému.

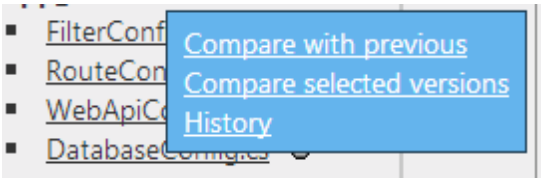
Compare -> Compare with previous version

History -> Compare selected versions

Graph -> History

Riešenie

Boli zmenené názvy aktívnych odkazov oproti predchádzajúcej verzie. Znázornené je na obrázku č. 46.



Obr. č. 46 Kontextové menu premenovanými odkazmi

9.11 Kontextové menu v AST strome a histórii

Cieľom tohto príbehu je poskytnutie kontextového menu naprieč celým systémom aby bola dodržaná konzistencia a zároveň bolo toto menu doplnené do stromu zobrazujúceho entity ako aj do histórie projektu.

Analýza

Problém sa bude týkať hlavne javascriptového kódu a kódu, ktorý je v CSHTML súboroch. Bude potrebné upraviť štruktúru HTML dokumentu tak, aby bolo riešenie univerzálne. Samotný javascriptový kód bude musieť byť tiež univerzálne využiteľný v rôznych CSHTML súboroch a nemôže byť závislý na jeho štruktúre.

Návrh

Samotný problém pozostáva z niekoľkých krokov:

1. Javascript - separácia kontextového menu do samostatného kódu.
2. CSHTML - úprava kódu, ktorý používal kontextové menu pred touto zmenou.
3. CSHTML - pridanie kontextového menu na požadované miesta.

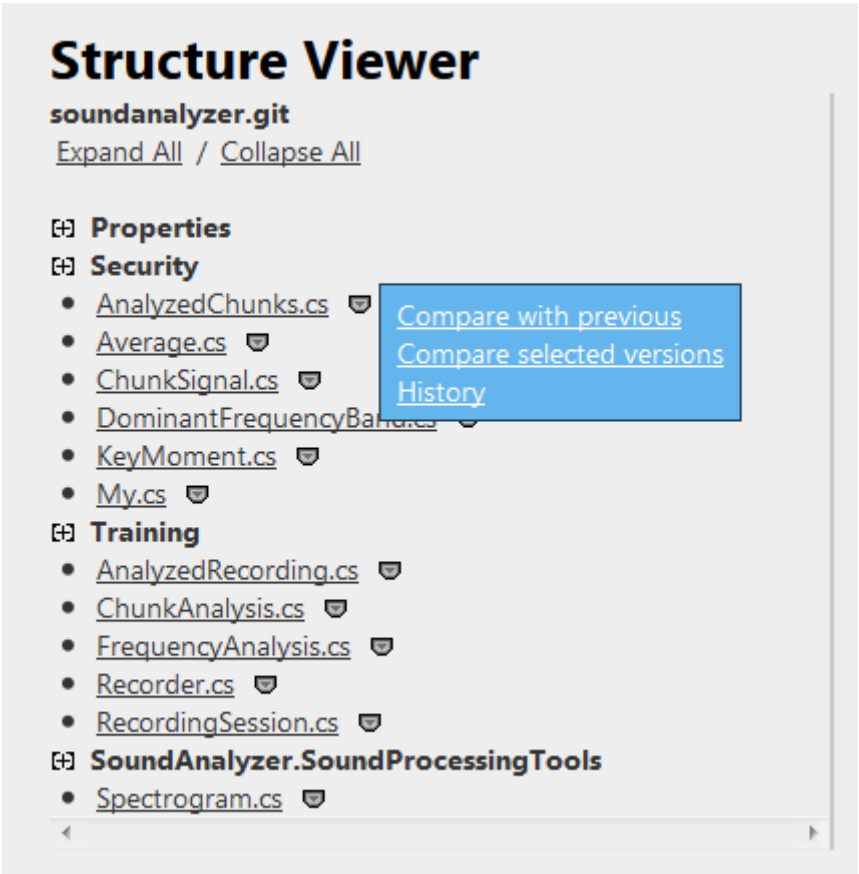
Riešenie

Samotné riešenie je potom založené na návrhu. Ako prvý bol upravený javascriptový kód zobrazujúci kontextové menu. Tento kód bol presunutý do samostatného súboru. Následne bol upravený CSHTML kód, ktorý kontextové menu využíval v minulosti. V poslednom kroku bolo pridané kontextové menu do zobrazovača entít ako aj do zobrazovača histórie repozitára (obr. 47).

```
<a href="#" class="menu-button"><img src="~/Images/li-toggle.png" /></a>
<div class="context-menu">
    Polozka 1<br />
    Polozka 2<br />
    Polozka 3... atd.
</div>
```

Obr. č. 47 Šablóna pre kontextové menu.

Samotné kontextové menu je potom zobrazené na obr. 48.



Obr. č. 48 Kontextové menu.

Testovanie

Testovanie prebiehalo skúšaním pôvodnej funkcionality a skúšaním novej funkcionality. Bolo testované zobrazovanie kontextového menu ako aj funkčnosť odkazov v ňom.

9.12 Výber iba dvoch verzií entít na porovnanie

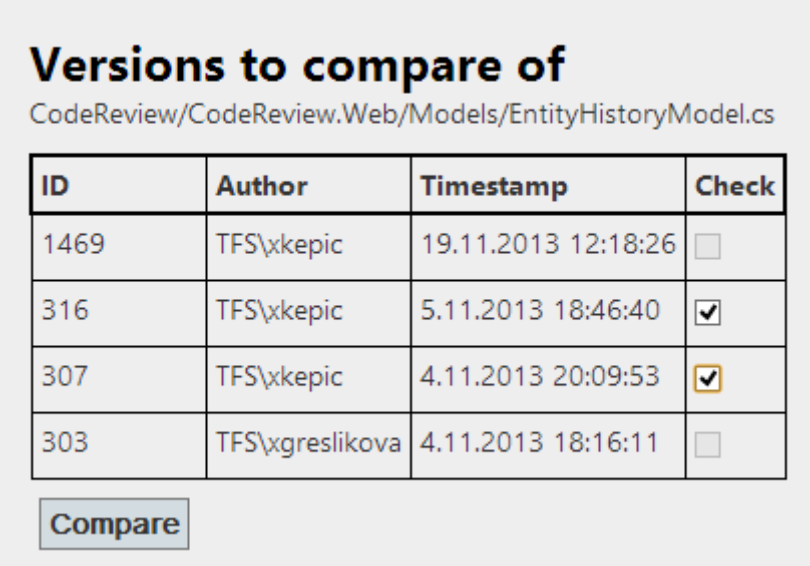
Používateľ požaduje povolenie výberu práve dvoch verzií entity pre porovnanie, aby mal istotu, ktoré dve verzie budú porovnávané, keď označí viac ako dve verzie.

Návrh

V tabuľke s históriou entity doteraz nebolo ošetrené zvolenie dvoch verzií entity pre porovnanie. Používateľ mohol označiť aj viaceré verzie na porovnanie, pričom sa porovnali iba dve verzie, ktoré boli najviac hore v tabuľke. Na to aby sa jednoznačne označili verzie, je možné použiť *JavaScript* funkciu, ktorá monitoruje počet vybraných verzii a ak sú označené dve verzie, zablokuje ostatné elementy typu *checkbox.*

Riešenie

Na obrázku č. 49 je uvedená tabuľka s históriou verzií, kde sú zablokované ďalšie verzie entity pre označenie.



Obr. č. 49 Tabuľka zablokovanými prvkami

9.13 Zachovanie hlavného pohľadu po zmene repozitára

Používateľ chce aby po zmene repozitára zostal zachovaný hlavný pohľad, aby mohol nerušenie pokračovať vo svojej práci.

Analýza

Pri prehliadaní obsahu repozitára sa dá kedykoľvek zmeniť repozitár na iný. V pôvodnom riešení sa vždy po zmene repozitára načítal obsah zmeneného repozitára a stránka sa presmerovala na domovskú stránku projektu. Budeme si musieť ukladať odkaz aktuálne navštívenej stránky a po zmene repozitára upraviť parametre odkazu a načítať stránku podľa uloženého odkazu.

Problém s ukladaním odkazu aktuálne navštívenej stránky nastane pri prezeraní obsahu vzťahujúceho sa na konkrétny repozitár (napr. pri prezeraní *changesetu*). V takomto prípade budeme musieť ukladať iba odkaz na predchádzajúcu stránku, ktorá nesúvisela priamo s konkrétnym repozitárom (napr. zobrazenie histórie odovzdaní, pretože história sa zobrazuje pre všetky repozitáre).

Riešenie

Pridali sme do metódy *Index* v *ProjectController* parameter *Url*, ktorý obsahuje aktuálne navštívenú stránku. Pokiaľ *Url* sa priamo nevzťahuje na žiaden konkrétny repozitár, upravíme v *Url* atribút *repositoryId*. Takto sa zobrazí aktuálna stránka so zmeneným repozitárom. *Url* adresu uložíme do *Session[„TempUrl“],* aby sme pri zmene repozitára na stránke vzťahujúcej sa ku konkrétnemu repozitáru vrátili na poslednú všeobecnú stránku. Na obrázku č. 50 je zobrazený kód, ktorý upravuje parametre v *Url*, a vkladá ich do *Session*.

```
Session["TempUrl"] = Url.Split('&').ToList().Select(x =>
                            {
                                   if (x.Contains("repositoryId"))
                                   {
                                          splited = x.Split('=');
                                          return splited[0] + "=" + repositoryId;
                                   }
                                   else return x;
                            }).Aggregate((m, n) => m + "&" + n);
```

Obr. č. 50 Zdrojový kód úpravy hodnoty atribútu v URL

9.14 SYSTÉMOVÁ SPRÁVA PROJEKTOV A REPOZITÁROV

Administrátor systému požaduje správu projektov a repozitárov aby mohol ku projektom repozitáre jednoducho pridávať a odoberať.

Analýza

Vstupné podmienky:

- Používateľ je administrátorom projektu
- Administrátor projektu je prihlásený

Výstupné podmienky:

- Zmena rozdelenia repozitárov v projekte

Účastníci: Administrátor projektu

Hlavný tok:

1. Administrátor projektu zvolí položku správa projektov.
2. Systém zobrazí všetky projekty, ktoré sú v databáze, označený je prvý projekt a k nemu sú zobrazené jeho repozitáre, ktoré sú k nemu priradené a nepriradené.
3. Administrátor vyberie repozitáre, ktoré chce pridať, prípadne odobrať od projektu a potvrdí výber.
4. Systém uloží zadané zmeny.
5. Prípad použitia končí.

9 Slniečko

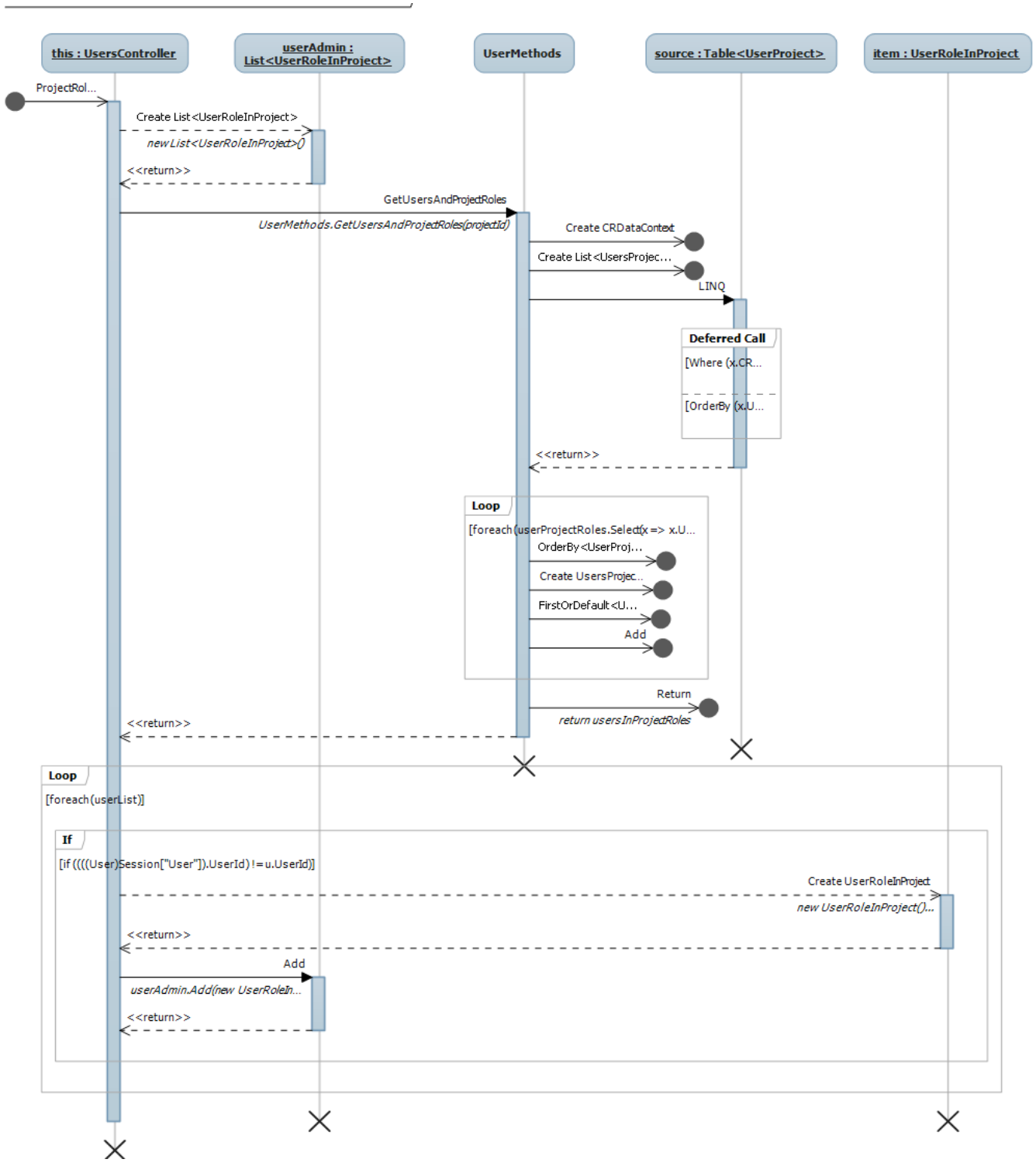
Alternatívne toky:

A1 Administrátor zvolí iný projekt

Tok sa aktivuje namiesto kroku 3.

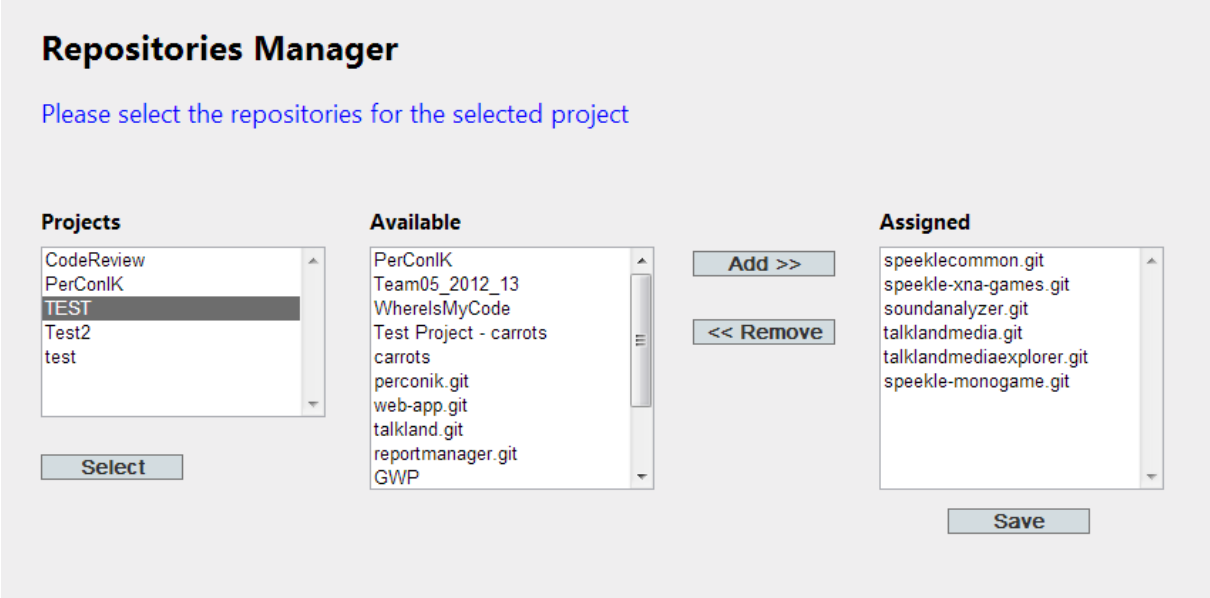
1. Administrátor projektu zvolí projekt, v ktorom chce zmeniť priradenie repozitárov a potvrdí výber.
2. Systém zobrazí repozitáre daného projektu, rozdelené na priradené a nepriradené.
3. Pokračuje sa krokom 3 hlavného toku.

Návrh
Sekvenčný diagram pre vytváranie modelu na zobrazenie je na obrázku č.51 :



Obr. č. 51 Sekvenčný diagram pre vytvorenie modelu

Administrátorovi projektu sú jednotlivé projekty a ich repozitáre zobrazené ako na obrázku, ako na obrázku č. 52. Používateľ si zvolí projekt s ktorým chce pracovať a pre vybraný projekt vyberie repozitáre. V časti Assigned sú repozitáre ktoré sú k danému projektu priradené a v časti available repozitáre ktoré projkt neobsahuje Po zvolení save je jeho výber uložený.



Obr. č. 52 Zobrazenie správy repozitárov projektov

Riešenie

Čísla changesetov: 738

Controller:

CodeReview.Web.Controllers.ProjectsManagerController

metódy:

[HttpGet]
public ActionResult Swapper()
[HttpPost]
public ActionResult Swapper(SwapperModel model)

Model:

CodeReview.Web.Models.UsersRolesModel.ProjectViewModel

View:

CodeReview.Web.Views.ProjectManager.Swapper

Testovanie

Test č.1:

Vstup: Zvolenie projektu, ktorý nebol zvolený implicitne „TEST“. Pridanie dvoch repozitárov z Assigned do Available (soundanalyzer.git, talklandmedia.git). Pridanie dvoch repozitárov z Available do Assigned (GWP, reportmanager.git). Ulozenie.

Výstup: Kontrola v databáze. Ulozenie prebehlo korektne. Administrátorovi sa zobrazí potvrdzujúca správa „Repositories were saved to current project“

Test č.2:

Vstup: Nezvolenie žiadneho repozitára a zvolenie Add.

Výstup: Administrátorovi sa objaví správa „No items were selected“.

9.15 Štatistiky značiek sú vždy počítané pre repozitár WhereIsMyCode

Chyba v kóde spôsobila že boli štatistiky počítané vždy z id repozitára 3.

Riešenie
Kód bol upravený aby počítal štatistiky pre zvolený repozitár.

Testovanie
Skúšali sme spustiť vyhľadanie štatistík pre rôzne repozitáre. Štatistiky sa zobrazovali správne.

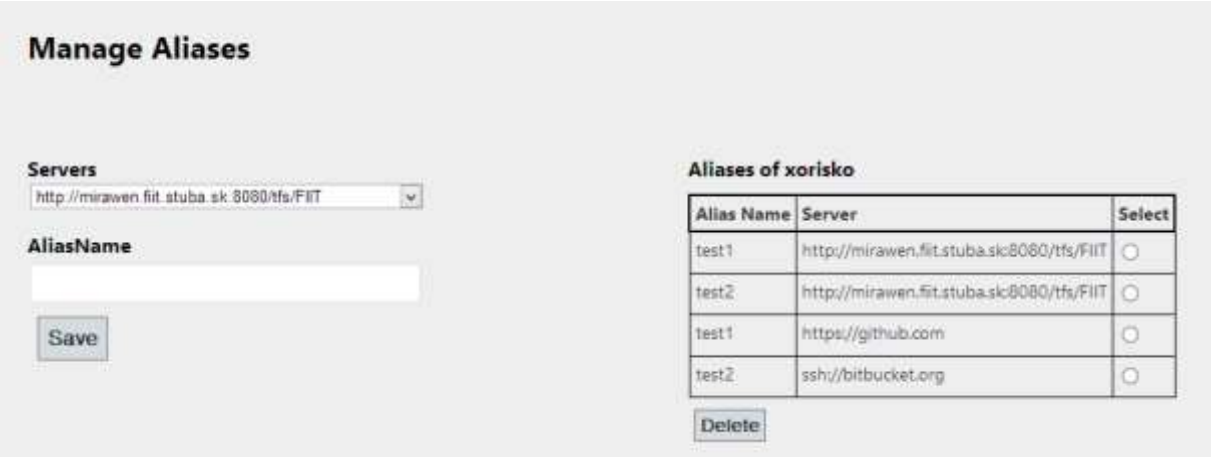
9.16 SPRÁVA ALIASOV POUŽÍVATEĽA

V šprinte Slniečko sa dokončuje používateľský príbeh správy aliasov používateľa. Vytvoria sa rozhrania pre jednoduchú prácu používateľa s aliasmi

Riešenie

- **Controller**
  - **UserAliasController –** riadi pridávanie a zmazanie Aliasov
- **Model**
  - **AliasModel –** model obsahuje aliasy prihláseného používateľa, ako aj zoznam serverov

Správa aliasov je prístupná z rozhrania *Account/Manage*, s kliknutím na odkaz *Manage Aliases*. Po kliknutí sa zobrazí rozhranie pre správu aliasov(Obrázok č. 53).



Obr. č. 53 Správa aliasov

View() ako parameter dostane model, ktorý obsahuje všetky aliasy aktuálne prihláseného používateľa a zoznam všetkých serverov. Zo zoznamu serverov sa naplní *DropDownList* naľavo, ktorý slúži na výber serveru, ku ktorej chceme pridať alias. Do *TextBox* je možné napísať meno aliasu. Kliknutím na tlačidlo *Save* sa uloží zadaný alias do databázy. V prípade úspešnej uložení sa zobrazí správa o pridaní záznamu do databázy a nový alias sa zobrazí vpravo v tabuľke aliasov. V prípade, že alias je v databáze, zobrazí sa chybová správa.

Odstránenie aliasov z databázy je umožnené v tabuľke aliasov pomocou elementov typu *RadioButton*. Po označení niektorého, záznam sa dá vymazať kliknutím na tlačidlo *Delete*. Po vymazaní z databázy sa zobrazí správa, a aktualizuje sa tabuľka aliasov používateľa.

9.17 ZÁKLADNÉ ŠTATISTIKY NAD PROJEKTOM

Používateľ chce vidieť na úvodnej stránke projektu základné štatistiky nad projektom, aby mal prehľad od stave riešenia projektu.

Analýza

Úloha zahŕňa :

- vytvorenie základnej infraštruktúry pre vytváranie štatistických nástrojov
- rozhranie pre nastavenie a spúšťanie štatistických modulov

Štatistiky majú zobrazovať:

- Počet odovzdaní
- Počet pridaných a odstránených neprázdnych riadkov kódu a ich celkový rozdiel
- Počty pridaných informačných značiek

Riešenie

Pre zvolený projekt bolo vytvorené rozhranie s možnosťou výberu typu zobrazovaných štatistík a času, v ktorom majú byť zisťované (obr. **54**). Štatistiky sumarizujú údaje pre všetky repozitáre v projekte.

Obr. č. 54 Rozhranie pre nastavenie a spúšťanie štatistických modulov

Obr. č. 55 Príklad zobrazenia štatistík

Ďalšie štatistiky budú priebežne dopĺňané podľa požiadaviek. Možnosť filtrovania napr. pre zobrazenie štatistík iba pre jeden repozitár budú rovnako pridané podľa požiadaviek neskôr.

Testovanie

Pri testovaní sme skúšali rôzne rozpätie dátumov a rôzne kombinácie pri výbere zobrazovaných značiek. Štatistiky sa zobrazovali správne.

9.18 USER S ADMINISTRÁTORSKÝMI PRÁVAMI SI MÔŽE ZRUŠIŤ ADMINISTRÁTORSKÉ PRÁVA

Pri nastavovaní administrátorských práv používateľom projektu, si mohol administrátorské práva odstrániť aj administrátor, ktorý bol aktuálne prihlásený.

Riešenie

Pri napĺňaní modelu sa zamedzilo vloženiu aktuálne prihláseného používateľa, a to podmienkou, kde *u* je vkladaní používateľ.

```
if ((((User)Session["User"]).UserId) != u.UserId)
```

K zmene došlo v „controlleri“ UsersController v metóde ProjectRolesManager.

9.19 STRÁNKOVANIE PRI NAČÍTANÍ AST PROJEKTOV DO DATABÁZY

Pri načítaní projektov z AST-RCS sa načítavalo len 20 projektov.

Riešenie

Implementovali sme stránkovanie pri načítaní projektov a načítali všetky projekty.

9.20 FILTRÁCIA HISTÓRIE ODOVZDANÍ - DOPRACOVANIE

User story Filtrácia histórie odovzdaní bol riešený v treťom šprinte s názvom Stonka. Úloha však nebola celkom doriešená, preto sme ju museli dopracovať . Opis vykonaných zmien poskytujeme v tomto šprinte.

Analýza

Aktuálne riešenie poskytuje filtráciu pre súborovú entitu a jednu vetvu projektu. Úpravami je potrebné dosiahnuť, aby bolo možno zobraziť vetvenie v prípade odovzdaní projektov a aplikovať filter aj na entitu zdrojového kódu. Autor grafu odovzdaní changesetov zatiaľ upravil model changesetov tak, že je možné namiesto dvoch konkrétnych predchodcov pridať list predchodcov a ten následne spracovať do grafu. Tým je vyriešený problém ak by po vymazaní mali väzby viac ako dvoch predchodcov.

Návrh

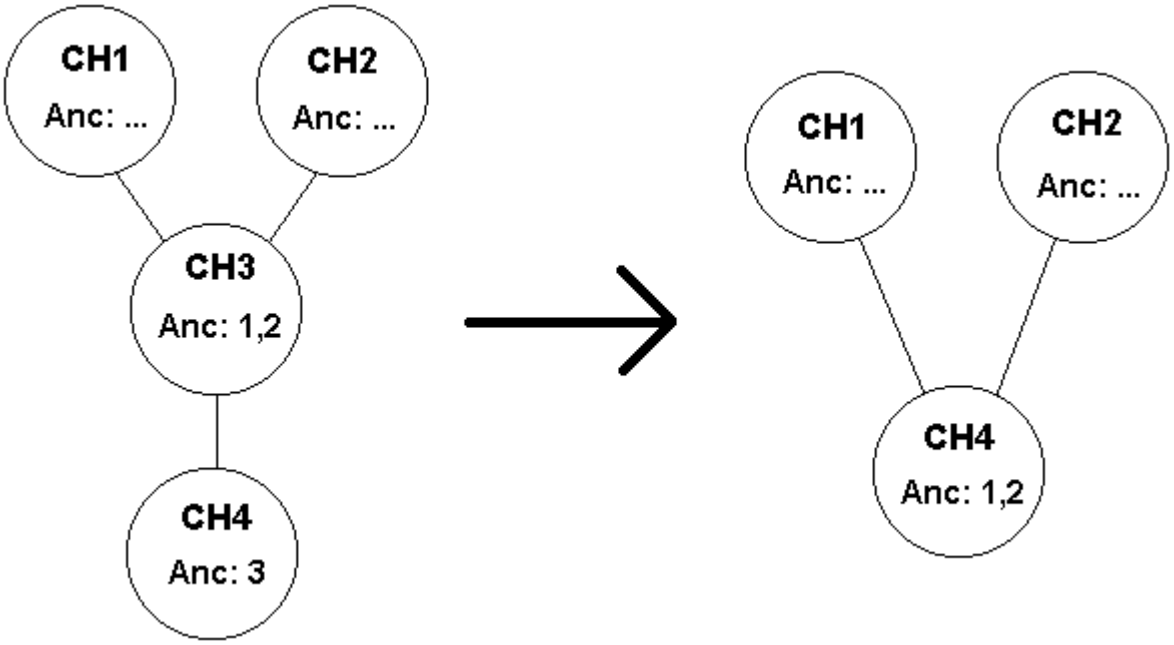
Pre vytvorenie filtra súborovej entity a kódovej entity je postup vykonania filtra rovnaký. Rozdielnym je len spôsob získania changesetov, kedy na nich boli vykonané zmeny. Pomocou ID verzie vyhľadáme changesety, avšak pre súborovú a kódovú entitu zavoláme im príslušne rozdielne funkcie získania changesetov.

Pre úpravu filtra a uloženie predchodcov upravíme mazaciu časť funkcie, ktorá odstráni changeset z listu jeho dieťaťa a upraví listy predchodcov, čím sa nebudeme odkazovať na neexistujúci changeset. Na záver upravíme aj funkciu pre tvorbu detí, ktorej výsledkom je v aktuálnom riešení tiež list a slúži na správne zobrazenie grafu.

Riešenie

Funkcie pre získanie changesetov súborovej alebo kódovej entity sme implementovali podľa ich špecifických volaní. Do parametrov pre volanie filtra pribudol parameter *type*, ktorým rozlíšime či ide o súborovú alebo kódovú entitu. Rozdiel voči predošlej verzií filtra je len v možností volania pre kódovú entitu, nielen pre súbor. Tým sme vyriešili prvý problém.

Druhým vyriešeným problémom bola úprava filtra changesetov. Ten sme vyriešili pomocou listov (zoznamov) a to tak, že po odstránení changesetu sme zaznamenali do jeho detí to, že ich predchodcami sú predchodcovia vymazaného changesetu. Toto riešenie môžeme pozorovať na obrázku č. 56.



Obr. č. 56: Úprava predchodcov (Ancestorov –Anc), po vymazaní changesetu s číslom 3

Zoznam predchodcov je v novej verzií formátu list, čiže zoznamu čísel. Okrem predchodcov vytvoríme pred dokončením celkového modelu na zobrazenie ešte pre každý changeset listy detí, ktoré sú súčasťou modelu a sú potrebné pre správne zobrazenie grafu. Hotový redukovaný zoznam changesetov následne odovzdáme na zobrazenie.

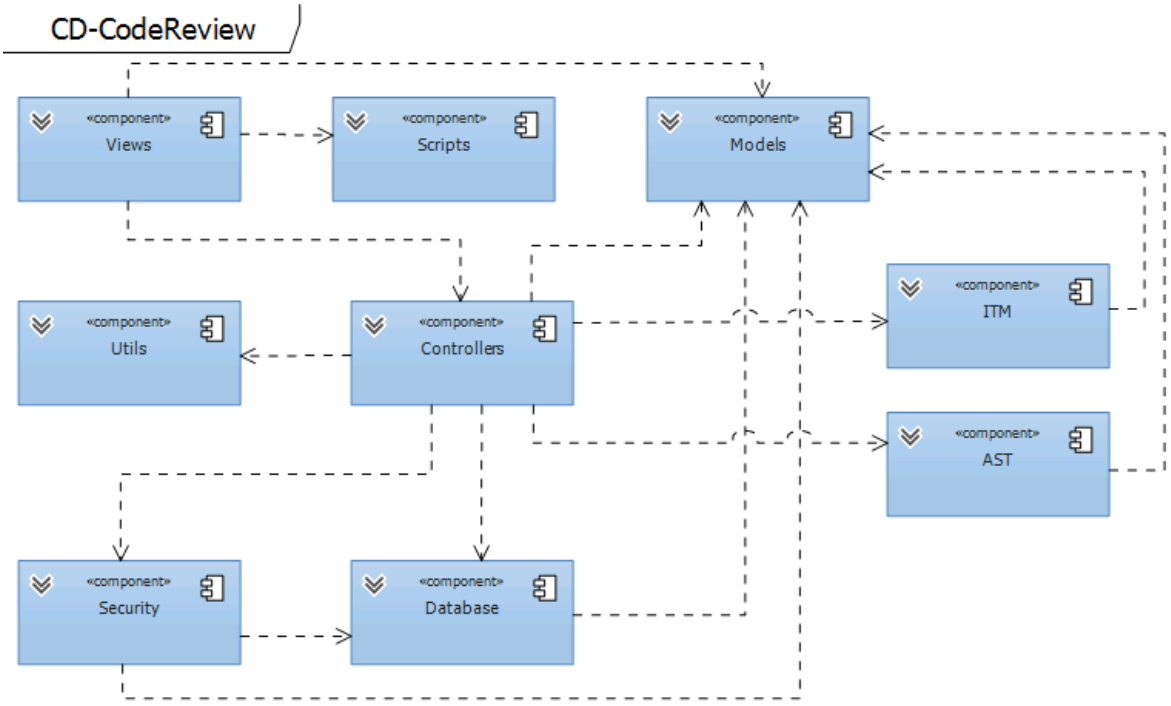
Testovanie

Testovanie bolo vykonané ako v predchádzajúcej verzii na základe vizuálnej kontroly a pozorovania, či sa vymazali správne changesety podľa histórie entity. Okrem vizuálneho testu je v aplikácií naprogramovaný jednotkový test, ktorý overuje, či sa správne zobrazila stránka grafu a zodpovedajúci počet changesetov. Testovanie prebehlo úspešne.

10 CELKOVÝ POHĽAD NA PROJEKT PO PIATOM ŠPRINTE

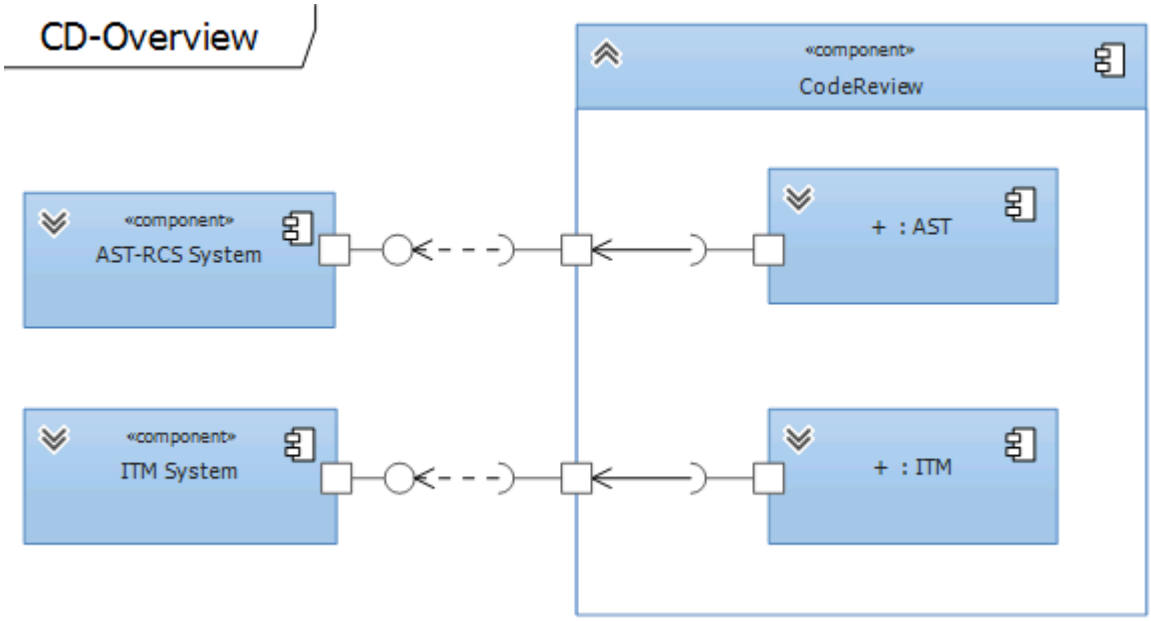
V tejto kapitole je zobrazený pohľad na softvér po 5 šprintoch. Sú tu uvedené „component diagramy“ a „use-case“ diagram systému ako celku. V „component diagramoch“ sú použite iba „<<use>> dependency“ vzťahy. Táto kapitola obsahuje aj dátový model databázy po 5 šprintoch.

„Component diagram“ celého systému je zobrazený na obrázku č. 57.



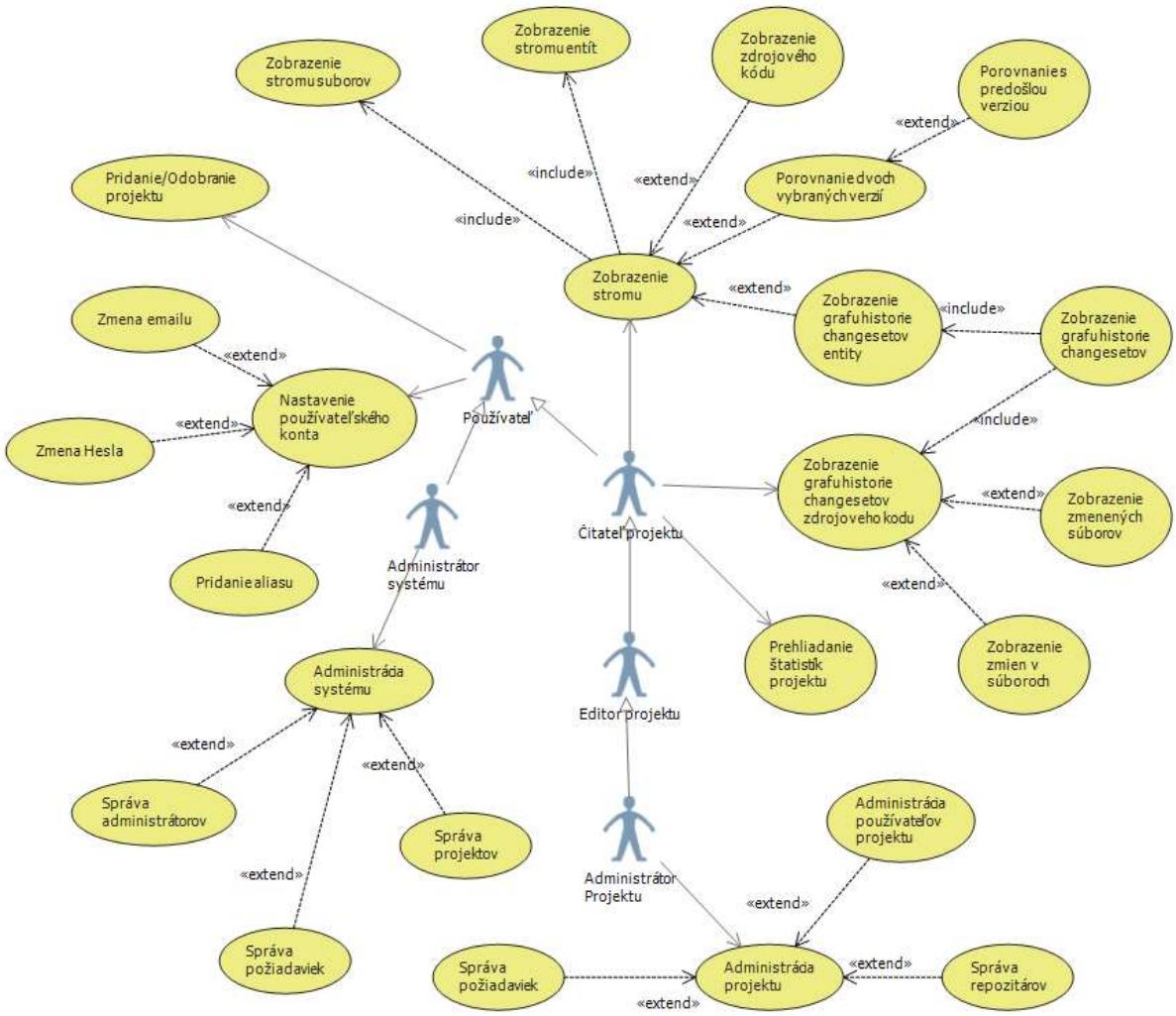
Obr. č. 57 "Component diagram" CodeReview

„Component diagram“ zobrazujúci komunikáciu medzi komponentmi CodeReview a inými systémami (ASTRCS a ITM) je zobrazený na obrázku č. 58. Komponenty CodeReview AST a ITM spracúvajú dáta zo systémov ASTRCS a ITM. Dáta sú ďalej posielané do „controllerov“.



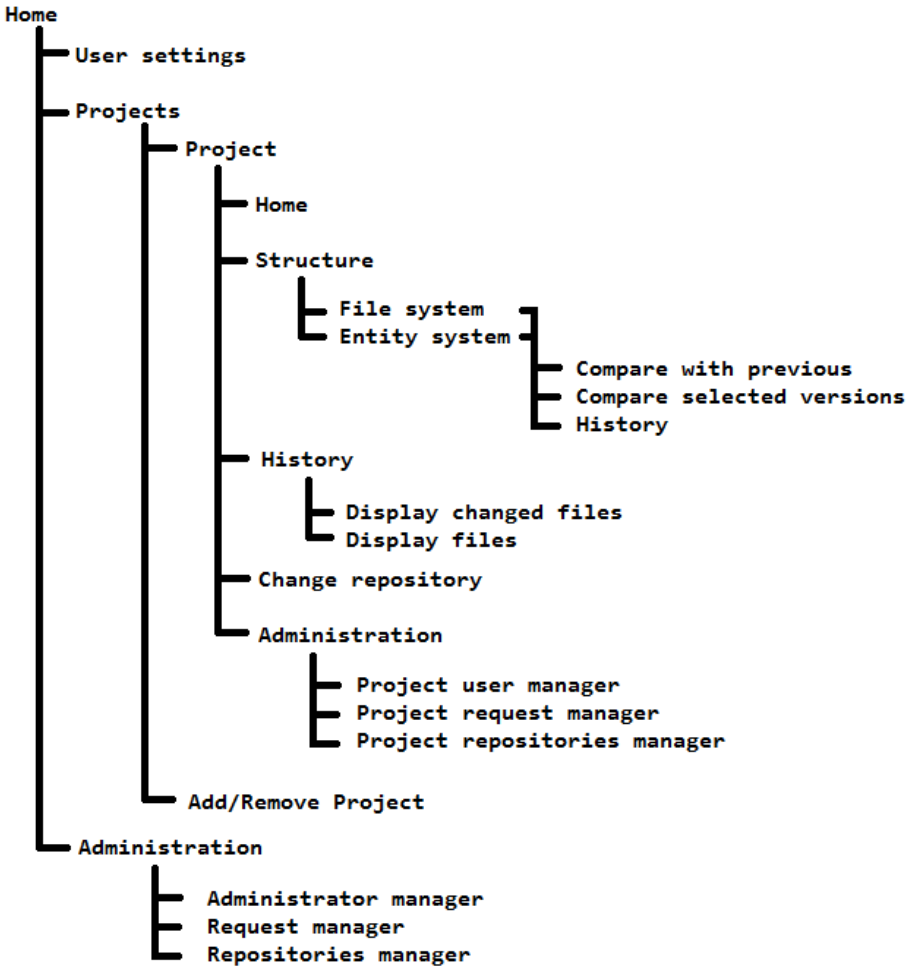
Obr. č. 58 Interakcia komponentov CodeReview s inými systémami

„Use Case diagram“ celého systému je zobrazený na obrázku č. 59.



Obr. č. 59 Use Case diagram projektu

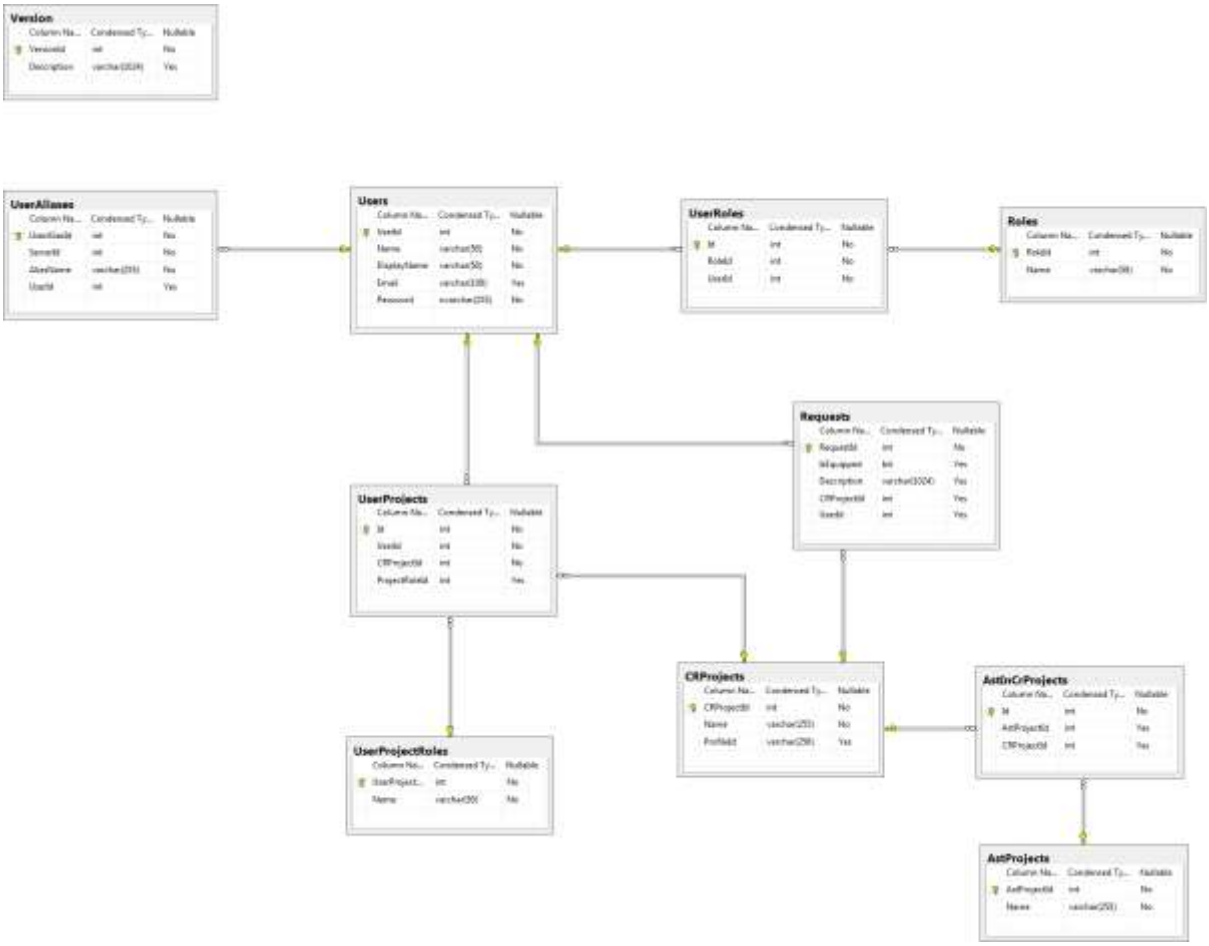
Mapa stránky projektu je zobrazená na obrázku č. 60.



Obr. č. 60 Mapa stránky projektu

Fyzický dátový model databázy je zobrazený na obrázku č. 61.

10 „Big Picture“



Obr. č. 61 Fyzický dátový model databázy

11 STANOVENIE GLOBÁLNYCH CIEĽOV PROJEKTU NA LETNÝ SEMESTER

V zimnom semestri sa nám podarilo vytvoriť dobré základy pre náš systém. Prvé úlohy boli jednoduchšieho charakteru. Naším cieľom bolo:

- Funkčné prihlásenie pre používateľa pomocou AIS loginu
- Vytvorenie kompletného databázového systému
- Správa používateľov
- Vytváranie projektov v systéme
- Správa repozitárov v systéme
- Zobrazenie zdrojového kódu a jeho zmien

Tieto úlohy sa nám podarilo veľmi dobre splniť pričom sme vytvorili dobré základy pre prácu na projekte v letnom semestri.

V letnom semestri sme sa budeme snažiť pridať nové funkcionality ako aj celý systém ukončiť tak, aby bol na konci letného semestra dobre použiteľný.

Medzi vybrané funkcionality, ktoré plánujeme pridať patria:

- Pridávanie značiek priamo cez prehliadač
- Vyhľadávanie v názvoch súborov
- Sprehľadnenie grafu zobrazujúceho jednotlivé vetvy projektu
- Systém notifikácii, ktorý bude informovať používateľov o pridaní značiek
- Funkčný prechod na konkrétny súbor alebo značku v projekte na základe linku
- Zoznam značiek v projekte a práca s týmto zoznamom
- Zmena dizajnu stránky

Na konci letného semestra by sme mali mať v rukách ucelený projekt, ktorý poskytuje danú funkcionalitu a pre koncového používateľa poskytuje hodnotu vo forme systému na jednoduché a prehľadné prehliadky zdrojových kódov v rôznych projektoch.

12 KVETINÁČÍK

Číslo šprintu: 6

Začiatok šprintu: 27.02.2014

Koniec šprintu: 13.03.2014

Príbehy:

- **Sprehľadnenie správy požiadaviek o priradenie repozitára k projektu**
- **Zjednotenie zobrazenia štruktúry repozitára so zobrazením súborov v odovzdaní**
- **Správa práv používateľov priradených k projektu**
- **Oddelenie štatistík od zoznamu používateľov projektu**
- **Informovanie o neexitujúcich štatistikách pre zvolený filter**
- **Zobrazenie informácie o prebiehajúcom načítaní kódu**
- **Umožniť prechod na zadanú URL**
- **Načítanie všetkých typov súborov**
- **Zobrazenie informácie o prebiehajúcom načítaní štatistík**
- **Začlenenie odoslania požiadavky o pripojenie repozitára pod jednotné rozhranie správy projektu**
- **Fulltextové vyhľadávanie nad menami súborových a kódových entít**
- **Zobrazenie aliasov pri štatistikách**
- **Zobrazenie zoznamu značiek**
- **Filtrovanie zoznamu značiek podľa ich typu**
- **Priradenie profilu používateľských značiek k projektu**
- **Presunutie na miesto v kóde, ktorému je priradená vybraná značka**
- **Informovanie o prebiehajúcom načítaní pri histórii**
- **Reset hesla**
- **Informovanie o prebiehajúcom načítaní štruktúry projektu**
- **Zdieľanie pomocou odkazu na zdrojový kód entity**

12.1 SPREHĽADNENIE SPRÁVY POŽIADAVIEK O PRIRADENIE REPOZITÁRA K PROJEKTU

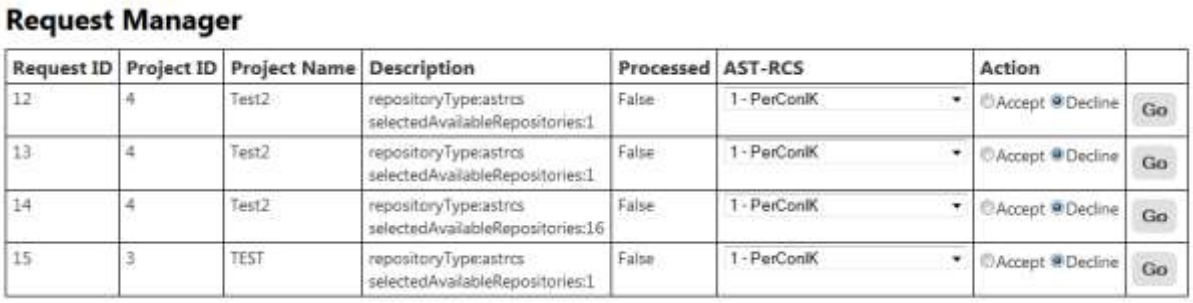
Aktuálne zobrazenie požiadaviek v administrátorskom menu nespĺňa požiadavky. Zákazník si želá vidieť viac detailov týkajúcich sa požiadavky.

Riešenie

Do rozhrania pridáme požadované informácie a samotnú tabuľku upravíme tak, aby spĺňala požadované kritéria. V rozhraní bolo vykonaných niekoľko zmien:

- Pridaný názov projektu, ku ktorému požiadavka patrí
- Popis už nezobrazuje prázdne polia
- Pri výbere projektu je ponúknutý zoznam namiesto textového políčka

Výsledok je zobrazený na obr. č. 62.



Obr. č. 62 Upravené rozhranie správy požiadaviek.

Testovanie

Testovanie prebehlo skúmaním zobrazených údajov a overením ich správnosti. Jednotlivé údaje boli korektné a samotné rozhranie zobrazovalo nové informácie.

12.2 ZJEDNOTENIE ZOBRAZENIA ŠTRUKTÚRY REPOZITÁRA SO ZOBRAZENÍM SÚBOROV V ODOVZDANÍ

Zákazník požaduje zjednotenie zobrazenia súborov v odovzdaní (možnosť "Display files") so zobrazením súborovej štruktúry projektu ("Structure->File system") aby mal sprístupnené rovnaké možnosti.

Riešenie

V súborovej štruktúre pre históriu odovzdaní bola pridaná možnosť „Display files“ rovnako, ako v súborovej štruktúre aktuálnej verzie. Bol použitý existujúci kód pre súborovú štruktúru aktuálnej verzie. Táto nekonzistencia vznikla rozdielnym spôsobom implementácie podobných častí rôznymi členmi tímu.

12.3 SPRÁVA PRÁV POUŽÍVATEĽOV PRIRADENÝCH K PROJEKTU

Zákazník požaduje umožnenie pridelenia práv používateľom, ktorí sú priradení k projektu, aby mohol administrátor projektu prideľovať potrebné práva jednotlivým používateľom.

Analýza

Vstupné podmienky:

- Používateľ je administrátorom projektu
- Administrátor projektu je prihlásený

Výstupné podmienky:

- Zmena práv používateľov v projekte

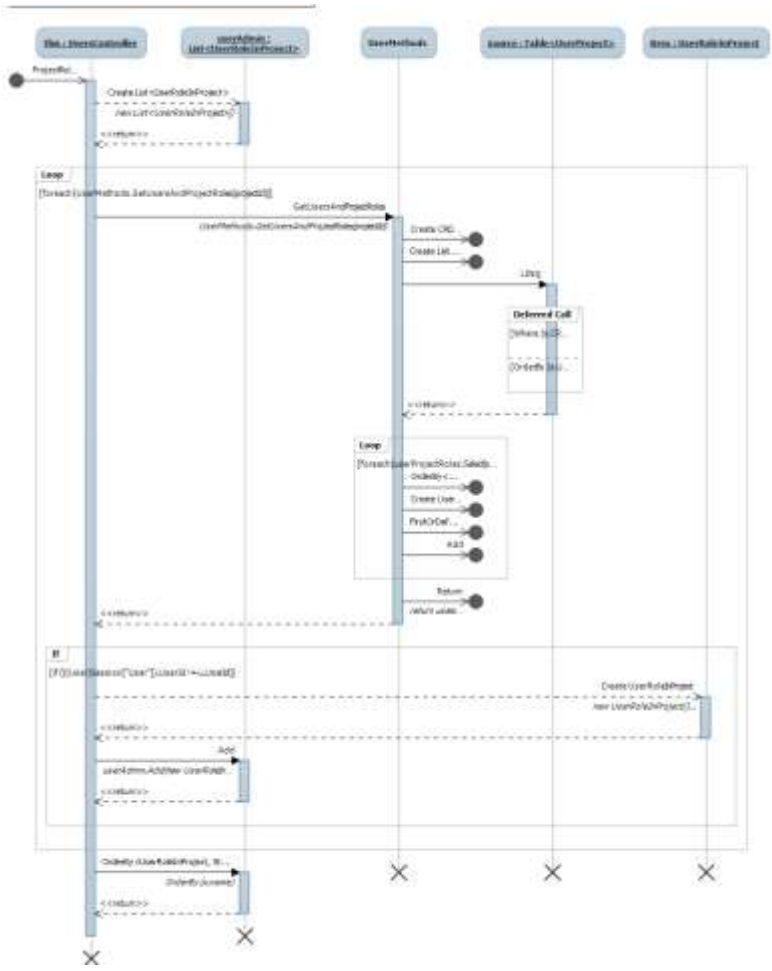
Účastníci: Administrátor projektu

Hlavný tok:

1. Administrátor projektu zvolí položku správa používateľov.
2. Systém zobrazí všetkých používateľov priradených k aktuálnemu projektu, ktorí sú v databáze. Vedľa daného používateľa je zvolená tá rola, ktorú má aktuálne priradenú.
3. Administrátor projektu vyberie pre používateľov roly, ktoré im chce priradiť a potvrdí výber.
4. Systém uloží zadané zmeny.
5. Prípad použitia končí.

Návrh

Sekvenčný diagram pre správu používateľov je na obrázku č.63:



Obr. č.63 Sekvenčný diagram pre správu používateľov

Administrátorovi projektu sú jednotliví používatelia a ich roly zobrazení ako na obrázku č. 64. Administrátor projektu si zvolí roly pre jednotlivých používateľov pričom každý z nich môže mať len jednu rolu. Jeho výber je automaticky uložený.

Project Roles Manager

User name	Project Administrator	Project Editor	Project Reader
aja	○	○	●
dakdnovy	○	○	●
kaledokd	○	○	●
test	●	○	○
vera	○	●	○

Obr. č.64 Zobrazenie správy používateľov projektov

Riešenie

Čísla changesetov: 967,971

Controller:

CodeReview.Web.Controllers.UsersController

metódy:

public ActionResult ProjectRolesManager(int projectId)

public ActionResult ChangeUserRoles(int projectId, int userId, int role)

Model:

CodeReview.Web.Models.UserRoleInProject

View:

CodeReview.Web.Views.Users. ProjectRolesManager

CodeReview.Web.Views.Users.EditorTemplates.UserRoleInProject

Testovanie

Vstup: Každému používateľovi bola zvolená iná rola ako mu bola pôvodne priradená. Potvrdenie výberu

Výstup: Kontrola v databáze. Uloženie prebehlo korektne.

12.4 ODDELENIE ŠTATISTÍK OD ZOZNAMU POUŽÍVATEĽOV PROJEKTU

Riešenie

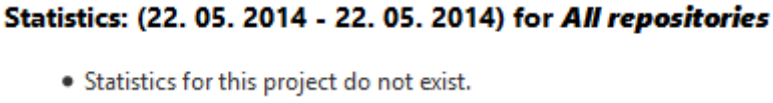
Doplnený grafický prvok pre sprehľadnenie používateľského rozhrania.

Obr. č. 65 Zobrazenie grafického prvku pre oddelenie používateľov od štatistík

12.5 INFORMOVANIE O NEEXITUJÚCICH ŠTATISTIKÁCH PRE ZVOLENÝ FILTER

Riešenie

Do zdrojového kódu bola pridaná podmienka pre zobrazenie informácie o neexistujúcich štatistikách.



Obr. č. 66 Príklad zobrazenie neexistujúcich štatistík

12.6 ZOBRAZENIE INFORMÁCIE O PREBIEHAJÚCOM NAČÍTANÍ KÓDU

Zákazník požaduje, aby načítanie súboru bolo doplnené o informáciu o samotnom procese načítania.

Analýza

Je potrebné preskúmať, akým spôsobom dosiahneme dynamickú zmenu obsahu HTML dokumentu po načítaní požadovaného súboru.

Návrh

Pre riešenie tohto problému použijeme ajaxové volanie, ktoré dynamicky zmení obsah formulára. Ešte predtým ako ho spustíme, zmeníme plochu, do ktorého sa má obsah súboru vložiť a doplníme ju o informáciu o vykonávanom procese.

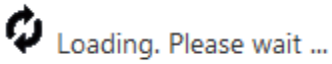
Riešenie

Samotné riešenie bolo veľmi jednoduché. Súbory sa už predtým načítavali asynchrónne. Bolo teda potrebné doplniť informáciu o tom, že samotný proces začal. K tomu využijeme možnosť pomocnej metódy Ajax.ActionLink, v ktorej sme pridali kód na obr. č. 67.

```
OnBegin = "insertSpinner();"
```

Obr. č. 67 Pridaný kód.

Funkcia insertSpinner() je definovaná v javascripte a táto funkcia iba zmení dokument pred zahájením asynchrónneho načítania súboru. Informácia o načítavaní je zobrazená na obr. č. 68.



Obr. č. 68 Doplnená informácia.

Testovanie

Testovanie prebehlo manuálne overením požadovanej funkcionality. Doplnený bol viac-menej iba kód javascriptu a celkové riešenie bolo triviálne.

12.7 UMOŽNIŤ PRECHOD NA ZADANÚ URL

Používateľ požaduje, aby mohol zadať zvolenú URL aj v prehliadači, ktorý nemá záznam o poslednom stave systému, aby mohol voľne a jednoducho pracovať aj na iných/cudzích pracovných staniciach.

Analýza

Pokiaľ nie je používateľ prihlásený a zadá URL adresu, na ktorú chce prejsť, systém mu zobrazí najskôr stránku s prihlásením. Po prihlásení do systému môže vložiť znovu URL adresu na ktorú chce prejsť a zobrazí sa mu. Takýto spôsob prechodu na zadanú URL ale nie je pohodlný, preto musíme umožniť používateľovi prechod na zadanú URL ihneď po prihlásení.

Návrh

Pri načítavaní *login* stránky musíme získať z prehliadača URL adresu a dočasne ju uložiť. Po úspešnom prihlásení je potrebné stránku presmerovať na túto URL adresu, namiesto pôvodne zobrazovanej úvodnej stránky.

Riešenie

Prechod na zadanú URL bol vyriešený použitím metódy *RedirectToLocal(returnURL);*. Ako parameter tejto funkcie sa vloží zadaná URL adresa z prehliadača, ktorú si dočasne uložíme v Session[„RedirectURL“]. Túto adresu po úspešnom prihlásení vložíme ako parameter do metódy *RedirectToLocal().*

12.8 NAČÍTAŤ VŠETKY TYPY SÚBOROV

Systém AST-RCS nedokáže ukladať obsah všetkých typov súborov. Z toho dôvodu bolo potrebné doplniť funkcionalitu pre rozoznávanie, ktoré súbory je možné zobraziť.

Analýza

Analyzovali sme typy súborov, pre ktoré systém nedokáže uložiť ich obsah. Pre takéto súbory sme vytvorili podmienku v zdrojovom kóde aby náš systém fungoval správne.

Riešenie

Do zdrojového kódu sme doplnili funkcionalitu pre zobrazenie informácie o súboroch, ktoré nie je možné zobraziť.

Obr. č. 69 Príklad zobrazenia informácie o nezobraziteľnom súbore

Testovanie

V našom systéme sme otestovali zobrazenie viacero typov súborov. Informácia sa zobrazovala správne.

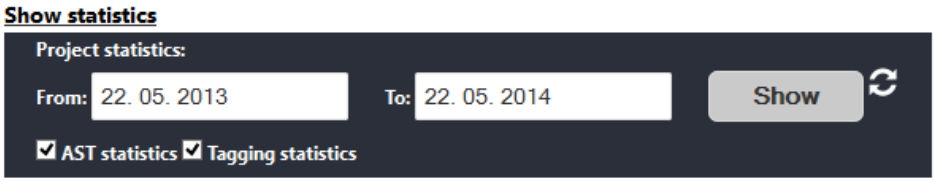
12.9 ZOBRAZENIE INFORMÁCIE O PREBIEHAJÚCOM NAČÍTANÍ ŠTATISTÍK

Návrh

Navrhli sme grafický prvok, ktorý zobrazuje priebeh načítavania štatistík. Prvok bol grafický upravený pre zosúladenie z dizajnom nášho systému.

Riešenie

Grafický "spinner" sa zobrazuje v priebehu načítavania štatistík zo systémov AST-RCS a ITM. Prvok sa zobrazuje na pravej strane panelu pre zobrazenie štatistík.



Obr. č. 70 Zobrazenie grafického prvky prebiehajúceho načítania štatistík

12.10 Začlenenie odoslania požiadavky o pripojenie repozitára pod jednotné rozhranie správy projektu

Zákazník požaduje, aby po odoslaní požiadavky na pridanie repozitára do projektu mohol túto požiadavku vidieť v systéme.

Riešenie

Samotné riešenie bolo veľmi jednoduché. Pre používateľa bolo pridané jednoduché rozhranie, ktoré zobrazuje všetky požiadavky viažuce sa na daný projekt. Používateľ vidí stav požiadavky, aj či bola vybavená. Samotné rozhranie je zobrazené na obr. č. 71.

Repository Requests

Request ID	Project ID	Description	Processed
5	3	repositoryType:astrcs selectedAvailableRepositories:3 tfsServerUrl: projectPath: tfsUsername: tfsPassword: gitRepositoryUrl: gitUsername: gitPassword:	True
15	3	repositoryType:astrcs selectedAvailableRepositories:1	False

Obr. č. 71 Rozhranie zobrazujúce požiadavky.

Testovanie

Testovanie prebehlo manuálne zobrazením rozhrania a kontrolou, či korektne zobrazuje všetky požiadavky, tak ako sa nachádzajú aj v samotnej databáze.

12.11 Fulltextové vyhľadávanie nad menami súborových a kódových entít

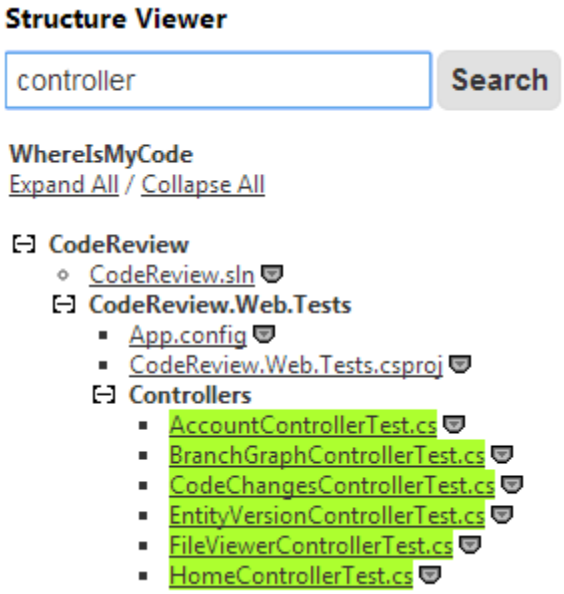
(Product Backlog Item 538): Používateľ požaduje možnosť fulltextového vyhľadávania nad súborových a kódových entít v projekte, aby mohol jednoduchšie nájsť hľadaný zdrojový kód. Toto hľadanie môže byť implementované pomocou filtra nad štruktúrou projektu.

Návrh

Táto funkcia sa dopĺňa do zobrazenia stromovej štruktúry súborov a kódových entít. Používateľovi po zadaní reťazca sa zobrazia výsledky vyhľadávania v stromovej štruktúre tak, že nájdené položky budú ofarbené a otvoria sa prislúchajúce uzly stromu. Táto funkcia bude implementovaná pomocou v jazyku *JavaScript*.

Riešenie

Funkcia vyhľadávania bola pridaná do súboru *dyntree.js* ktoré obsahuje všetky metódy pracujúce so zobrazením stromovej štruktúry zdrojových kódov a kódových entít.



Obr. č. 72 Vyhľadávanie nad menami súborových a kódových entít

Na obrázku č. 72 je znázornená výsledok vyhľadávanie pre reťazec „controller“. Vyhľadávanie akceptuje aj čiastkové slová a nerozlišuje malé a veľké písmená.

Testovanie

Tento príbeh bol testovaný vizuálne. Výsledky hľadania boli porovnané manuálne menami zdrojových kódov a kódových entít.

12.12 Zobrazenie aliasov pri štatistikách

Používateľ chce aby sa jeho štatistiky zobrazovali spojené pre všetky aliasi v systémoch, v ktorých sú jeho repozitáre uložené.

Návrh

Z už implementovanej funkcionality pre pridávanie aliasov sme použili všetky aliasi používateľov z databázy. Tieto aliasi boli následne v našom systéme nahradené jedným používateľským menom.

Riešenie

Implementovali sme metódu pre zisťovanie aliasov z databázy a ich následné spojenie pod jedným menom. Každý používateľ, ktorý má v našom systéme jeho aliasi zo systémov, v ktorých sú jeho repozitáre uložené, vidí štatistiky spojené pod jedným používateľským menom.

12.13 Zobrazenie zoznamu značiek

Používateľ chce zobraziť na jednom mieste všetky značky, ktoré sa nachádzajú v projekte, konkrétne vo zvolenom repozitári.

Analýza

Hlavnou úlohou je získanie zoznamu značiek a jeho následne zobrazenie. Zoznam môžeme získať dvoma spôsobmi:

1. Použitím služby ktorá vráti zoznam značiek v repozitári
2. Vlastnými metódami prehľadávajúcimi súborovú štruktúru projektu

Prvý spôsob jednoducho vráti zoznam značiek zo zadaného repozitáru. Druhý spôsob je náročnejší na implementáciu, avšak aktuálne máme implementovanú metódu na získanie značiek zo súboru.

Návrh

V našom riešení sme sa rozhodli neimplementovať poskytnutú službu, ale vlastný spôsob získavania značiek. Na to budeme potrebovať vytvoriť funkcie na prehľadávanie súborovej štruktúry aktuálneho repozitáru a zoznam obsahujúci nový model, do ktorého budeme zaznamenávať značky. Vytvorený zoznam následne zobrazíme na novej stránke obsahujúcej zoznam značiek.

Riešenie

Keďže načítavanie značiek budeme vykonávať jednotlivo po súboroch, navrhli sme model zobrazený na obrázku č. 73, z ktorého vytvoríme zoznam.

```
public class FileTagsModel
{
    public string FileName { get; set; }
    public int FileId { get; set; }
    public List<TagsModel> Tags = new List<TagsModel>();
}
```

Obr. č. 73: Model pre ukladanie zoznamu značiek v súbore

Tento model obsahuje meno súboru, Id súboru a zoznam značiek, ktoré sa nachádzajú v súbore. Táto štruktúra je vhodná hlavne pre zobrazenie jednotlivých značiek po súboroch a následne jednoduchší prístup k ním do súborov, keďže máme uložené informácie o súbore, v ktorom sa nachádzajú.

Na vytvorenie zoznamu značiek v súboroch sme vytvorili novú metódu. Tá pomocou ďalšej vytvorenej metódy na postupne prehľadávanie súborovej štruktúry repozitáru vyhľadá značky. Vyhľadávanie prebieha takým spôsobom, že sa postupne prechádzajú súbory v repozitári a pomocou už implementovanej metódy na získavanie značiek v súbore získavame jednotlivé značky. Ak súbor obsahuje nejaké značky, vytvorí sa nová položka zoznamu s názvom a Id súboru, do ktorej sa vloží zoznam značiek príslušného súboru (viď model).

Vytvorený zoznam značiek podľa súborov je následne potrebné zobraziť. V menu projektu sme vytvorili v položke ,,Structure“ novú možnosť - ,,Project tags“. Po kliknutí sa zobrazí zoznam značiek vo zvolenom repozitári. Zoznam pozostáva z jednotlivých súborov - názvu súboru, ktorý slúži aj ako odkaz a z jednotlivých značiek, ktoré sú zobrazené iba ako text. Odkaz pomocou názvu súboru nás presmeruje na konkrétny súbor v stromovej štruktúre súborov.

Testovanie

Testovanie prebiehalo ručne na základe zobrazenia rôznych súborov a pozorovania, či v zozname značiek boli zobrazené všetky značky príslušného súboru. Pozorovali sme, že zoznam značiek korektne zobrazil značky z jednotlivých súboroch v repozitári.

12.14 Filtrovanie zoznamu značiek podľa ich typu

Používateľ chce mať možnosť filtrovať zoznam značiek pomocou výberu konkrétneho typu značky.

Analýza

Aktuálne zobrazujeme zoznam všetkých značiek nachádzajúcich sa v určitom repozitári. Filter značiek podľa typu sa bude zaoberať naším získaným zoznamom, pričom z neho vyberie iba požadovaný typ značky.

Návrh

Filtrovanie značiek vykonáme nad získaným zoznamom značiek po procese, keď už budeme mať získané všetky značky repozitáru.

Na stránke zoznamu značiek implementujeme možnosť výberu typu značky.

Riešenie

Prvým krokom pri implementácii filtra bolo jeho vloženie na stránku. Filter sme vytvorili ako HTML prvok *select list* (*dropdown list*), ktorý obsahuje nami zadané hodnoty možných značiek a jednu špeciálnu možnosť pre zobrazenie všetkých značiek. Pri prvom načítaní stránky sa zobrazia všetky značky a filter. Vo filtri následne môžeme vybrať nami zvolený typ značky z poskytnutých možností. Po výbere a potvrdení sa stránka načíta nanovo, pričom sa spustí funkcia filtra.

Pri načítaní stránky sa opäť získa celý zoznam značiek, avšak po kliknutí na filter sa navyše aplikuje metóda vykonávajúca filtráciu. Tá prebehne na našom zozname súborov so značkami a vytvorí nový zoznam, ktorý obsahuje iba značky zvoleného typu. Funkcia vráti prefiltrovaný zoznam, ktorý sa už klasicky zobrazí.

Testovanie

Pri testovaní sme manuálne skúšali filter a pozorovali zmeny. Filter sa aplikoval korektne. Na stránke však nastala drobná chyba. Keďže sa načítala nanovo, filter značiek sa aplikoval avšak v HTML prvku s výberom značky - *select liste*, neostala nami zvolená možnosť, ale ,,defaultne“ zadané zobrazenie všetkých typov značiek. Tento problém bude potrebné vyriešiť.

12.15 Priradenie profilu používateľských značiek k projektu

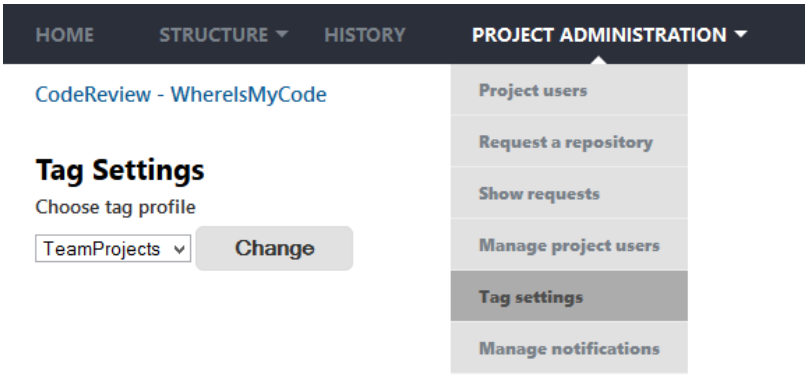
Pridávanie značiek je možné z použitím rôznych profilov značiek. Tieto profily obsahujú rôzne typy značiek. Používateľ chce mať možnosť si vybrať z dostupných profilov značiek.

Analýza

Analyzovali sme akým spôsobom systém ITM ukladá profily značiek. Použitím dostupných metód sme určili spôsob pre načítanie dostupných profilov značiek.

Riešenie

Do systému sme doplnili možnosť pre zmenu používaného profilu značiek. Zvolený profil sa ukladá do databázy.



Obr. č. 74 Výber používaného profilu značiek

12.16 PRESUNUTIE NA MIESTO V KÓDE, KTORÉMU JE PRIRADENÁ VYBRANÁ ZNAČKA

(Product Backlog Item 70): Používateľ potrebuje možnosť presunúť sa na miesto v kóde, na ktorom sa nachádza značka zobrazená v zozname značiek.

Návrh

Táto funkcia bude vykonaná zo zoznamu (*Project Tags v menu*) značiek pri zvolení zobrazenia konkrétnej značky. Kliknutím na odkaz pre značku sa zobrazí zdrojový kód súboru obsahujúci zvolenú značku tak, že sa vykoná presunutie v zdrojovom kóde na polohu značky a značka bude zvýraznená žltým rámikom v kóde.

Riešenie

Obr. č. 75 Presun na značku

Na obrázku č. 75 je zobrazený vykonaný presun na značku v kóde so zvýraznením značky. Funkcia príbehu bola implementovaná v jazyku *JavaScript*.

Testovanie

Testovanie prebiehalo manuálne preklikaním zoznamu značiek. Funkcionalita presunu fungovala správne.

12.17 Informovanie o prebiehajúcom načítaní pri histórii

Pre zobrazenie stromu v histórii, je potrebne zmeniť doterajší View "TreeView" na "StructureView".

Návrh

Zobrazenie stromu v histórií je potrebné zjednotiť so zobrazením stromu v štruktúre aktuálneho odovzdania. Tento problém by vyriešilo „rozkúskovanie“ zobrazenej stránky na parciálne stránky, aby bolo výsledné zobrazenie zložené z viacerých komponentov. Hlavný komponent, ktorý zobrazuje jednotlivé komponenty bude obsahovať všetky používané javascripty, ako aj javascript „spinner“, ktorý informuje o prebiehajúcom načítaní údajov. Tento hlavný komponent bude používaný pre všetky stránky, v ktorých sa požaduje zobrazenie stromu súborov aj zobrazenie zdrojového kódu.

Riešenie

Vytvorili sme stránku *StructureView*, ktorá obsahuje všetky javascripty používané v zobrazení štruktúry súborov. Ostatné komponenty (strom súborov, strom entít, strom súborov v histórií, zobrazenie kódu, zobrazenie zmien v kóde) sme implementovali ako parciálne komponenty („partial views“). Týmto spôsobom sa vyriešilo informovanie o prebiehajúcom načítaní v histórií a zároveň sa zjednotilo zobrazenie stromu a zdrojového kódu.

12.18 Reset hesla

Zákazník požaduje, aby používateľovi bolo umožnené resetovať heslo, keď heslo zabudne.

Analýza

Vstupné podmienky:

- Používateľ už niekedy v minulosti bol prihlásený v systéme
- Používateľ zle zadal svoje heslo

Výstupné podmienky:

- Používateľovi bolo resetované heslo na heslo z AIS

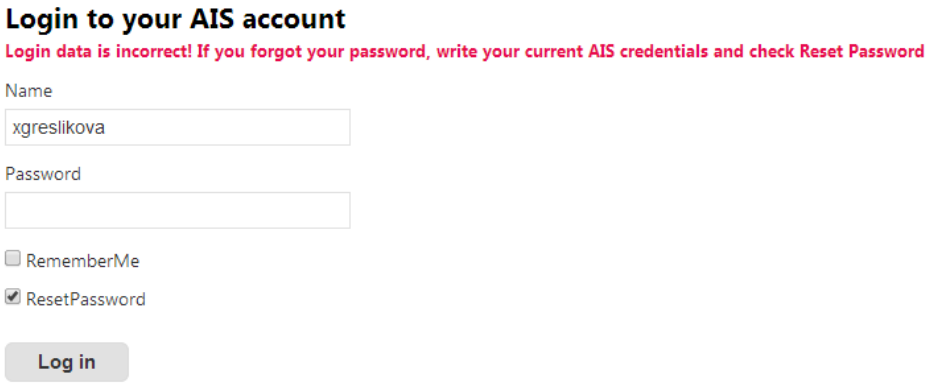
Účastníci: Neprihlásený používateľ

Hlavný tok:

1. Používateľ zvolí pri prihlásení položku „Reset password“.
2. Systém vyzve používateľa aby pre reset hesla zadal jeho heslo do systému AIS.
3. Používateľ zadá svoje heslo do systému AIS.
4. Systém resetuje heslo používateľa na heslo z AIS a uloží ho v databáze a prihlási používateľa do systému.
5. Prípad použitia končí.

Návrh

Používateľovi sa po zadaní zlého hesla ponúkne možnosť heslo resetovať. Po jeho zvolení je používateľ upozornení na to, že jeho heslo bude resetované po zadaní prihlasovacieho hesla do AIS ako na obrázku č. 76.



Obr. č. 76 Zobrazenie resetu hesla

Riešenie

Čísla changesetov: 1005

Controller:

CodeReview.Web.Controllers. AccountController

metódy:

public ActionResult Login(User user)

Model:

CodeReview.Web.Models.User

View:

CodeReview.Web.Views.Account.Login

Testovanie

Vstup: Používateľ pri prihlásení zadal zlé heslo. Následne zvolil položku „ResetPassword“ a zadal svoje heslo do AIS. Po tom, čo sa úspešne prihlásil sa používateľ odhlásil a pri ďalšom prihlásení zadá resetované heslo.

Výstup: Prihlásenie prebehlo korektne a heslo v databáze bolo resetované na pôvodné heslo do AIS.

12.19 Informovanie o prebiehajúcom načítaní štruktúry projektu

Zákazník požaduje, aby bol informovaný o tom, keď sa načíta stromová štruktúra projektu. Tento proces môže trvať aj niekoľko sekúnd, preto sa vyžaduje doplnenie informácie o vykonávaní.

Analýza

Podobne ako v úlohe o načítavaní súboru, je potrebné preskúmať, akým spôsobom dosiahneme dynamickú zmenu obsahu HTML dokumentu po načítaní požadovaného súboru.

Návrh

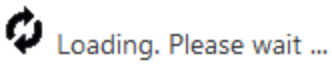
Samotná úloha je trochu komplexnejšia. Bude potrebné upraviť aktuálny kód tak, aby štruktúra stromu bola v samostatnom dokumente. Tento dokument vložíme dynamicky do iného dokumentu, ktorý zatiaľ zobrazuje iba informáciu o vykonávanej akcii.

Riešenie

Riešenie spočívalo v niekoľkých krokoch.

1. Prenesenie vytvorenia stromovej štruktúry do samostatnej akcie v kontroléry. Výsledkom je vrátené parciálne view, ktoré vložíme do iného dokumentu.
2. Napísanie javascriptu, ktorý po vykreslení základného dokumentu zavolá ajaxové volanie, ktoré doplní informáciu o strome.

Pred samotným zobrazením stromovej štruktúry je zobrazená informácia (obr. č. 77).



Obr. č. 77 Doplnená informácia.

Testovanie

Testovanie prebehlo manuálne. Najprv sa overil, či upravený pôvodný kód funguje korektne, potom bola otestovaná aj funkčnosť javascriptu. Testovanie bolo úspešné.

12.20 Zdieľanie pomocou odkazu na zdrojový kód entity

Pre zdieľanie kódov bolo je nutné vytvoriť zdolateľný odkaz pri zobrazení súborov projektu.

Analýza

V tomto prípade prichádzajú do úvahy dve riešenia tohto problému. Prvou je úprava linku súboru tak aby v ňom bol zahrnutý aj zobrazený súbor. Druhou je vytvorenie tlačidla, ktorý vytvorí odkaz na zobrazenie iba daného zdieľaného súboru.

Riešenie

Implementovaná bola prvá metóda keďže kvôli zdieľaniu iba samotného súboru by bol kontext tohto súboru zanedbaný. Druhým dôvodom bola aj jednoduchšia implementácia prvého spôsobu.

Úprava podoby linku sa vykonala pomocou parametrov JavaScriptu. Samotná implementácia je jednoduchá, v prípade že sa v linku nachádzajú parametre zavolá sa AJAX, ktorý načíta súbor pomocou implementovaného controlera.

Táto funkcia sa volá pri udalosti po načítaní stránky v „dyntree.js“ .

Testovanie

Prebehlo manuálne na rôznych súboroch a s rôznymi parametrami.

13 PRVÝ LÍSTOK

Číslo šprintu: 7

Začiatok šprintu: 13.03.2014

Koniec šprintu: 27.03.2014

Príbehy:

- **Po prihlásení do systému je zobrazený javascript**
- **Zobrazenie zmien v kóde v zobrazení histórie**
- **Úprava dizajnu správy práv používateľov priradených k projektu**
- **Pridanie odkazov na značky zo zoznamu značiek**
- **Zoznam správ**
- **Správa notifikácií**
- **Vytvorenie a zmazanie notifikácie o vytvorení značky**
- **Refactoring štatistík a stromovej štruktúry**
- **Refactoring histórie a zobrazenia zdrojových kódov**
- **Zvýšenie prehľadnosti rozhrania**
- **Úprava URL odkazov na súbory**
- **Vytváranie značiek**
- **Synchrónne scrollovanie pri zobrazení zmien v súbore**

13.1 PO PRIHLÁSENÍ DO SYSTÉMU JE ZOBRAZENÝ JAVASCRIPT

Tento problém so zobrazením javascriptu je spôsobený prechodom na zadanú URL, ale vyskytuje sa iba pri prihlásení na server a v náhodných intervaloch, preto sa tento problém nepodarilo odstrániť.

13.2 ZOBRAZENIE ZMIEN V KÓDE V ZOBRAZENÍ HISTÓRIE

Pri zobrazovaní histórie zdrojových kódov sa zobrazuje aktuálna verzia kódu, v ktorej nie sú zvýraznené zmeny z predchádzajúcej verzie zdrojového kódu.

Riešenie

Komponent na zobrazenie zdrojového kódu sa nahradil komponentom (už existujúcim) pre zobrazenie zmien medzi aktuálnou a predchádzajúcou verziou zdrojového kódu.

13.3 ÚPRAVA DIZAJNU SPRÁVY PRÁV POUŽÍVATEĽOV PRIRADENÝCH K PROJEKTU

Zákazník požaduje aby tabuľka správy práv používateľov bola zobrazená inak.

Návrh

Odstránenie prebytočných riadkov a rozdelenie tabuľky. Pridanie hlavičky v tabuľke. Zmena tagu <td> na tag <th>, ktorý predstavuje hlavičku tabuľky.

Riešenie

Čísla changesetov: 1156

View:

CodeReview.Web.Views.Users. ProjectRolesManager

CodeReview.Web.Views.Users.EditorTemplates. UserRoleInProject

13.4 PRIDANIE ODKAZOV NA ZNAČKY ZO ZOZNAMU ZNAČIEK

Používateľ chce mať možnosť pristúpiť k značke priamo zo zoznamu značiek pomocou odkazu na konkrétny súbor, v ktorom bude cez parameter určený konkrétny riadok.

Analýza

V aktuálnom riešení je v zozname možné použiť odkaz na súbor, ktorý obsahuje značky. Používateľ však požaduje mať prístup k jednotlivým značkám osobitne, pričom sa pri zobrazení súboru ,,nascrolluje“ určitý riadok so značkou.

Návrh

Tak, ako je vytvorený odkaz na súbor, bude potrebné vytvoriť tento odkaz pre jednotlivé značky. Rozdielom však je odovzdanie parametra s riadkom, na ktorom sa nachádza zvolená značka.

Riešenie

Pri zobrazení zoznamu značiek sme ku každej značky pridali odkaz formou textu ,,Link“, ktorý sa odkazuje na zvolenú značku. Odkaz má rovnaký formát ako odkaz na súbor, rozdielom je pridaný parameter s číslom riadku na ktorý je potrebné sa ,,nascrollovať“. Autor zobrazovania súborov upravil ich zobrazovanie tak, že je možné prostredníctvom vhodne vytvoreného odkazu odovzdať

parameter s číslom riadku, ktorý sa následne spracuje a zobrazí sa ,,nascrollovaná“ pozícia so značkou.

Testovanie

Odkazy na súbory prebiehali správne. Vždy sa zobrazil požadovaný súbor, v ktorom sa nachádzala značka. Jediným problémom bolo v niektorých prípadoch správne zobrazenie riadku so značkou, ktoré však nebolo spôsobené naším postupom, ale formátom ukladania značiek na serveri. Problémom teda nebolo ,,nascrollovanie“ alebo odkaz na značku, ale získané číslo riadku značky.

13.5 Zoznam správ

Používateľ požaduje zobrazenie správ z notifikácií, pričom chce vedieť počet neprečítaných notifikácií, vidieť zoznam posledných 10 správ, mať možnosť načítať ďalších 10 správ pomocou tlačidla "Show more" a mať možnosť si notifikáciu prečítať.

Analýza

V našom projekte je aktuálne vytvorený systém pre notifikácie používateľov. Jednou z jeho častí je zoznam správ notifikácií, ktorý je uložený v databáze. Tento zoznam je potrebné zobraziť prostredníctvom webového rozhrania.

Návrh

Zoznam správ notifikácií sa bude zobrazovať v dvoch zoznamoch. Jeden bude vytvorený ako klikateľná položka v menu, ktorá bude zobrazovať 10 správ. V tejto voľbe bude okrem zobrazenia správ zahrnutá aj možnosť zobrazenia ďalších správ, pričom tie sa už načítajú na samostatnej stránke.

Po kliknutí na odkaz v menu alebo pri zobrazení všetkých značiek zobrazíme údaje konkrétnej značky.

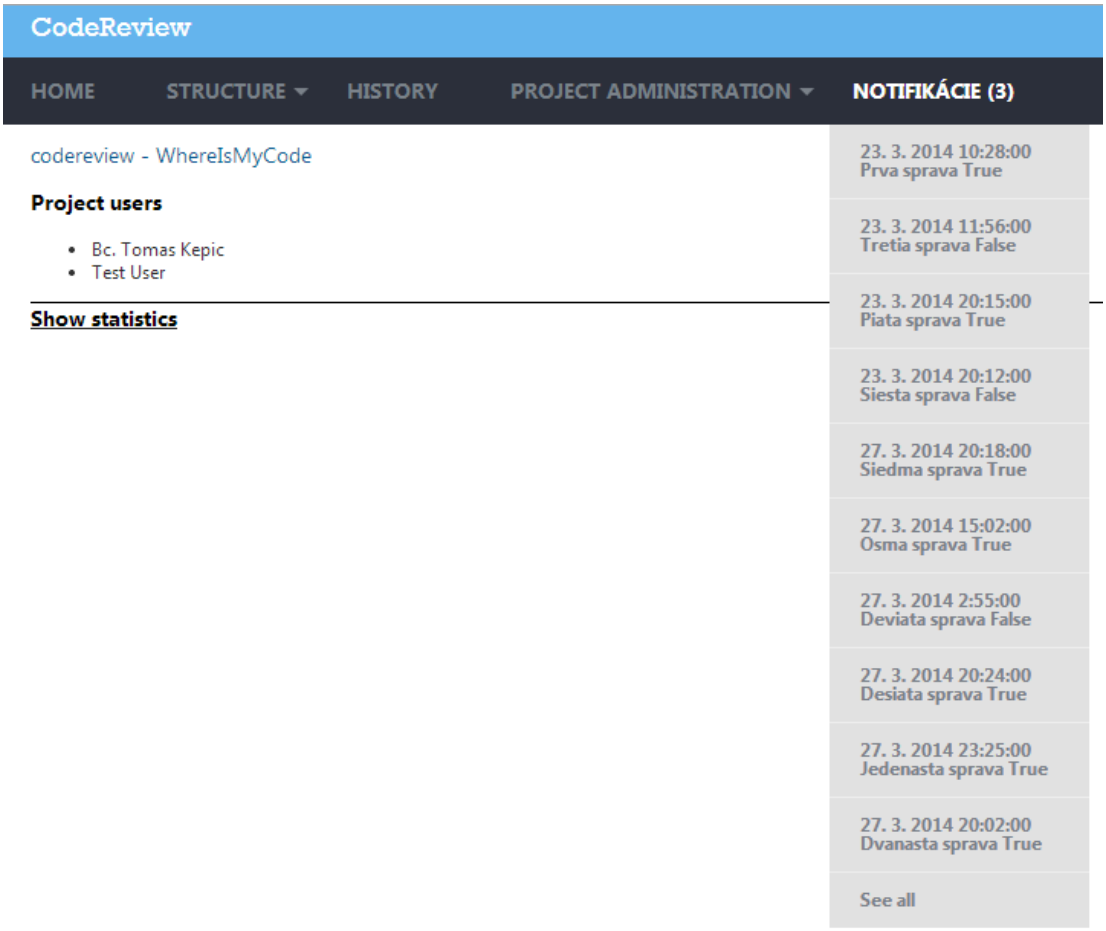
Dôležitou informáciou, ktorú zahrnieme do menu, je zobrazenie počtu neprečítaných notifikácií na základe údaju o prečítaní uloženom pri každej notifikácií. Po zobrazení neprečítanej notifikácie sa tento údaj v notifikácií zmení na prečítaný.

Riešenie

Prvou úlohou je vytvorenie klikateľnej položky v menu, čiže zobrazenie 10 správ notifikácií a počtu neprečítaných notifikácií. Vytvorili sme metódu na získanie správ z databázy. Tá získava údaje na základe prihláseného používateľa (ktorému patria notifikácie) a vybraného projektu.

Druhou úlohou a metódou je zistenie počtu neprečítaných notifikácií.

Získané údaje následne zobrazujeme v položke v menu, čiže do základného *view*, ktoré obsahuje menu na úrovni projektu sme pridali položku so správami notifikácií. Nové menu sme naplnili získaným zoznamom správ, pričom sme zvolili obmedzenie na 10 správ. Pomocou funkcie na zistenie počtu neprečítaných správ sme zistili ich počet a číslo sme uviedli za položku v menu. Po kliknutí a rozbalení zobrazíme údaje jednotlivých správ: dátum, krátky popis správy a údaj o jej prečítaní. Rozbalenú položku menu môžeme pozorovať na obrázku č. 78.



Obr. č. 78: Výpis 10 správ notifikácií v menu

Po kliknutí na správu sa nám zobrazia jej údaje, ktorými sú dátum, krátky a dlhý popis a stav prečítania. Tieto údaje sa momentálne zobrazujú na samostatnej novej stránke. Po kliknutí na správu, ktorá ešte nebola prečítaná, sa zmení údaj v databáze o prečítaní. Taktiež sa pri novom načítaní menu zmení údaj o počte neprečítaných notifikácií.

Položka v menu po rozkliknutí obsahuje okrem správ aj možnosť ,,See all“, ktorá slúži na zobrazenie všetkých správ na samostatnej stránke. Tieto správy nie sú zobrazené všetky naraz, ale je zobrazených 10 správ. Na konci výpisu je tlačidlo ,,Show more“, pomocou ktorého je možné načítať zoznam ďalších 10 správ, ktoré sa pridajú k aktuálne zobrazenému zoznamu. Pri správe zobrazujeme dátum, krátky popis a stav prečítania. Po kliknutí na ,,Show details“ môžeme vidieť okrem týchto údajov aj dlhý popis k značke. To sa však zobrazuje na novej stránke.

Testovanie

Testovanie prebiehalo manuálne, pričom sme porovnávali stav v databáze a zobrazené položky v menu, prípadne na vytvorenej stránke kde sú vypísané všetky značky. Pri testovaní sme pozorovali nedostatok, ktorým je neautomatické odpočítanie počtu neprečítaných správ po kliknutí na neprečítanú správu. Aktuálne je potrebné znovu načítať celú stránku aby sa aktualizovalo menu. Tento problém bude potrebné vyriešiť.

13.6 SPRÁVA NOTIFIKÁCIÍ

Zákazník požaduje, aby bol pripravený systém pre notifikácie. Rozhranie umožní používateľovi pridať notifikáciu rôzneho typu, poskytne možnosť zmazať daný typ notifikácie, a tiež možnosť zvoliť si, či o danom type notifikácie chce byť používateľ notifikovaný alebo nie.

Analýza

Vstupné podmienky:

- Používateľ je prihlásený v systéme
- Používateľ si zvolil projekt

Výstupné podmienky:

- Používateľ si nastaví svoje notifikácie

Účastníci: Administrátor projektu

Hlavný tok:

1. Používateľ si zvolí položku správy notifikácií.
2. Systém zobrazí všetky doterajšie notifikácie ktoré boli k projektu vytvorené (s ikonami na ich odstránenie, editovanie alebo zvolenie o správe emailom) spolu s možnosťou pridať novú notifikáciu.
3. Používateľ si zvolí možnosť zmazať notifikáciu.
4. Systém zmaže danú notifikáciu z databázy.
5. Prípad použitia končí.

Alternatívne toky:

A1 Administrátor zvolí editáciu notifikácie

Tok sa aktivuje namiesto kroku 3.

6. Používateľ zvolí možnosť editovať notifikáciu.
7. Systém zobrazí vybranú notifikáciu, typ tejto notifikácie, popis a notifikovanie o danej notifikácií.
8. Používateľ zmení vybrané atribúty notifikácie. Zmenu potvrdí.
9. Systém uloží zadané zmeny do databázy.
10. Prípad použitia končí.

A2 Administrátor zmení možnosť notifikovania emailom

Tok sa aktivuje namiesto kroku 3.

11. Používateľ zvolí ikonu notifikácie emailom.
12. Systém zmení súčasný stav notifikovania na opačný a ikona sa zmení aby používateľ bol o zmene informovaný. Zmena sa uloží v databáze.
13. Prípad použitia končí.

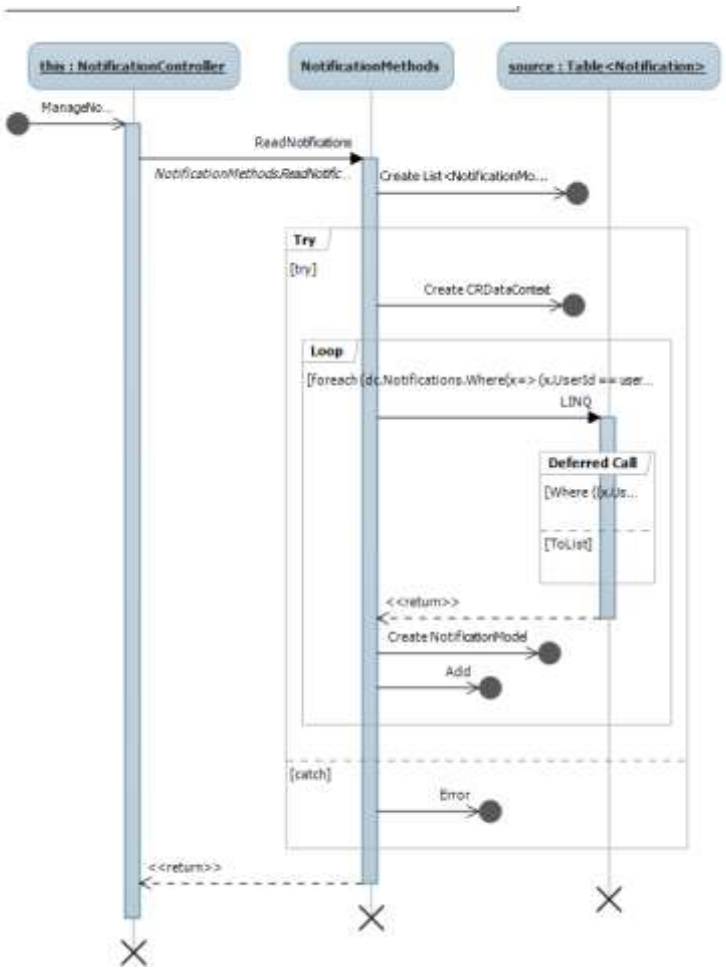
A3 Administrátor zvolí možnosť pridania novej notifikácie

Tok sa aktivuje namiesto kroku 3.

14. Používateľ zvolí možnosť pridať notifikáciu.
15. Systém prejde na časť vytvorenie notifikácie popísanej v časti 12.7.
16. Prípad použitia končí.

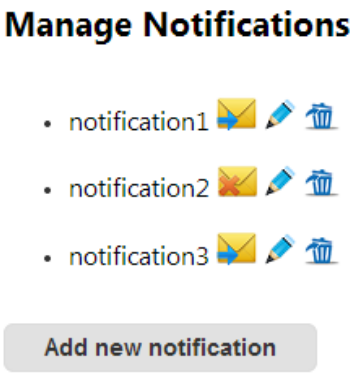
Návrh

Sekvenčný diagram na zobrazenie notifikácií je na obrázku č. 79 :



Obr. č. 79 Sekvenčný diagram na zobrazenie notifikácií

Používateľovi systému sa zobrazia jednotlivé notifikácia s popisom. Používateľ si môže následne zvoliť ikonu na základe toho či chce danú notifikáciu zmazať, editovať alebo zmeniť notifikovanie. Notifikácie sú zobrazené ako na obrázku č. 80. V spodnej časti má používateľ možnosť pridať notifikáciu.



Obr. č. 80 Zobrazenie správy notifikácií

Riešenie

Čísla changesetov: 1080, 1129, 1156, 1186, 1191

Controller:

CodeReview.Web.Controllers.NotificationController

metódy:

public ActionResult ManageNotification(int projectId)

[HttpPost]

public ActionResult ManageNotification(IEnumerable<NotificationModel> model)

public ActionResult DeleteNotification(int notificationId, int projectId)

public ActionResult EditEmailNotification(int notificationId, bool selected)

Model:

CodeReview.Web.Models.NotificationModel

View:

CodeReview.Web.Views.Notification.ManageNotification

Testovanie

Test č.1:

Vstup: Zvolenie Editácie notifikácie, zmena parametrov notifikácie, uloženie notifikácie.

Výstup: Kontrola korektného zobrazenia zmenenej notifikácie.

Test č.2:

Vstup: Zmazanie „Notifikacie1“.

Výstup: kontrola v databáze, kontrola pri obnovení stránky, daná notifikácia sa nenachádza medzi ostatnými notifikáciami.

Test č.3:

Vstup: Zmazanie notifikovania o notifikácií z nezasielania správy na zasielanie správy a naopak.

Výstup: kontrola v databáze, kontrola pri obnovení stránky.

13.7 VYTVORENIE A ZMAZANIE NOTIFIKÁCIE O VYTVORENÍ ZNAČKY

Zákazník vyžaduje funkcionalitu, ktorá umožní v systéme prijímať notifikácie o pridaných značkách v projekte. Používateľ je prostredníctvom stránky informovaný o pridanej značke a táto informácia je zobrazená vo forme notifikačnej správy. Úlohou je vytvoriť systém, kde si používateľ môže zadefinovať, aké typy značiek ho zaujímajú a o týchto značkách bude informovaný.

Analýza

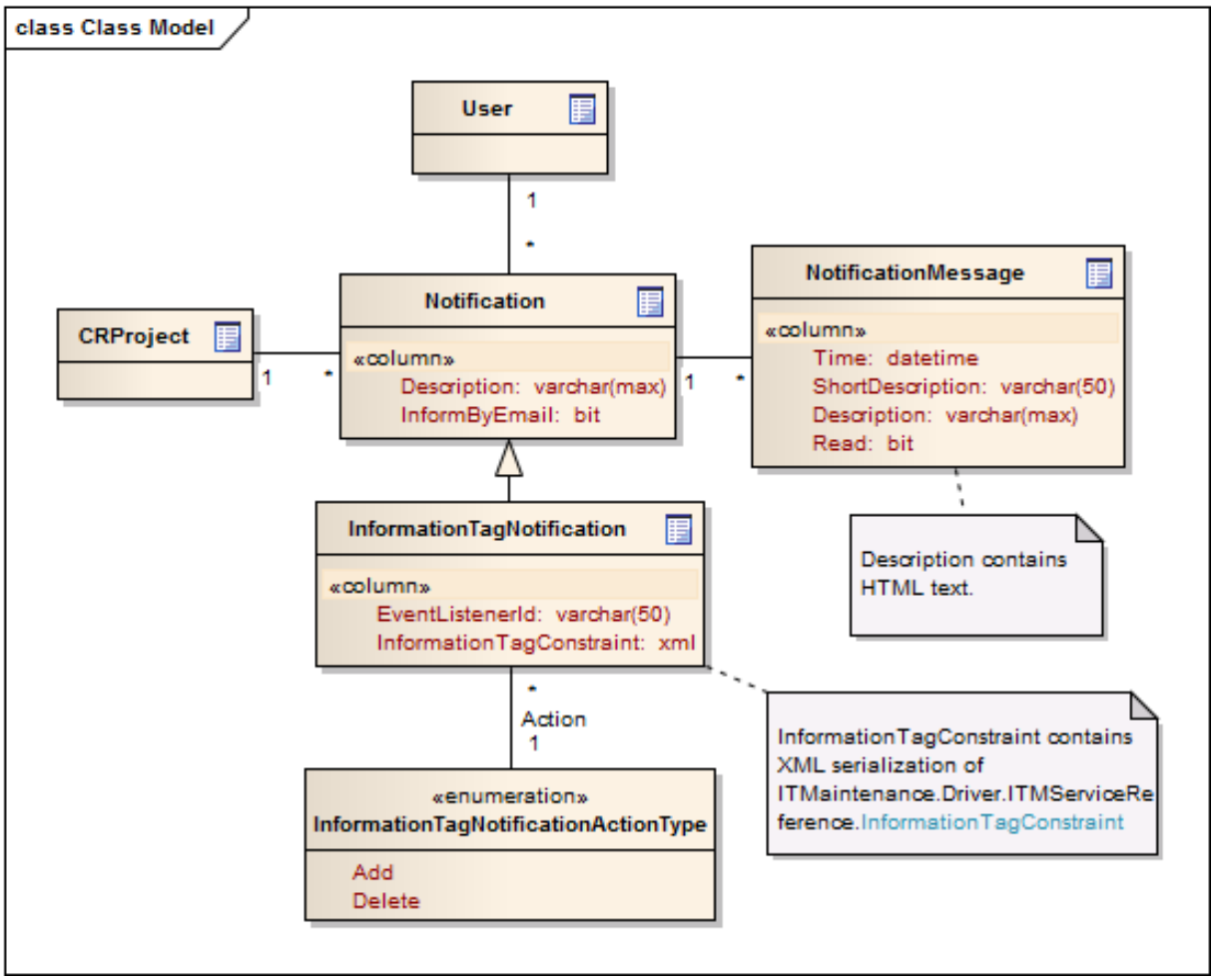
Táto úloha sa skladá s viacerých krokov a bude potrebné vedieť, ako sa majú jednotlivé kroky vykonať. Okrem toho musí systém spolupracovať so systémom značiek, ktorý bude zároveň posielať notifikačné správy do našej aplikácie.

Návrh

Samotný návrh sa skladá z niekoľkých krokov:

1. Rozhranie na vytvorenie a úpravu notifikácie – typ značky, opis a či má byť informovaný mailom.
2. Registračný mechanizmus, ktorý kontaktuje ITM server. Ako odpoveď potom prijme unikátny identifikátor registrovanej notifikácie.
3. Služba na prijímanie notifikačných správ.
4. Mechanizmus na odregistrovanie notifikácie.

Kvôli tejto úlohe bude taktiež upraviť databázu na základe nasledujúceho návrhu (obr. č. 81).



Obr. č. 81 Dátový model notifikačného systému.

Riešenie

Riešenie bolo vyhotovené podľa návrhu. Najprv bolo vytvorené základné rozhranie, ktoré obsahuje formulár na vytvorenie alebo úpravu značky. Tento formulár je vytvorený automaticky na základe modelu.

Po spracovaní sú hodnoty odoslané na ITM server, ktorý ich prijme a vráti ako odpoveď unikátne ID zaregistrovanej notifikácie.

Prijatá notifikačná správa je spracovaná v REST službe, ktorá prijíma správy z ITM servera. Samotná služba, resp. to čo sa vykonáva s notifikačnou správou je potom predmetom úlohy v ďalšom šprinte.

Testovanie

Testovanie bude prebiehať až v nasledujúcom šprinte, kedy sa dokončí kompletne REST služba na prijímanie notifikačných správ.

13.8 Refactoring štatistík a stromovej štruktúry

Z dôvodu sprehľadnenia kódu bolo potrebné vykonať sme vykonali refactoring. Kontrola prebehla zdrojovom kóde týkajúcom sa stromovej štruktúry a štatistík.

Riešenie

V zdrojovom kóde boli vykonané nasledujúce úkony:

- Dopĺňanie komentárov
- Úprava názvov
- Úprava zložitých štruktúr v kóde
- Oprava nájdených chýb

13.9 Refactoring histórie a zobrazenia zdrojových kódov

(Product Backlog Item 774): Úlohou bolo spraviť refactoring nad históriou a zobrazením zdrojových kódov vrátane back-and, front-end a testov.

Návrh

Na vykonávanie príbehu je potrebné preštudovať zdrojové súbory pracujúce históriou a zobrazením zdrojových kódov, identifikovať pachy v týchto kódoch a odstrániť ich. Pachy sa majú nájsť v súbore *EntityVersionController*. Ďalej sa majú opraviť zobrazenia pracujúce so súborom *EntityVersionController* a je potrebné implementovať test pre správnu funkcionalitu spomenutého kontroléra.

Riešenie

V *EntityVersionController* bol identifikovaný pach duplicitnej metódy. Tento pach bol odstránený. Zobrazenia, ktoré vrátia metódy kontroléra boli prerobené na parciálne zobrazenia. Všetky sa zobrazia vedľa stromovej štruktúry súborových a zdrojových entít. Ďalej bol vytvorený test pre overenie správnej funkcionality kontroléra.

13.10 Zvýšenie prehľadnosti rozhrania

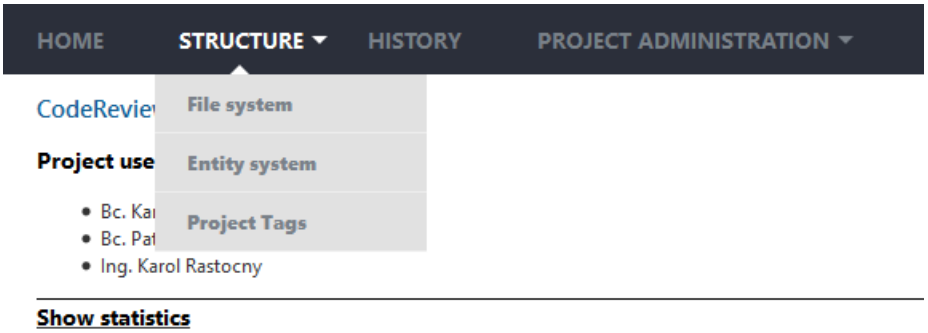
Z dôvodu nízkej prehľadnosti používateľského rozhrania sme upravili rozhranie pre jednoduchšie použitie.

Návrh

Zmenili sme názvy v menu na obrázkové tlačidlá. Pridané dynamické menu projektu. Zmena rozmiestnenia niektorých prvkov.

Riešenie

Obr. č. 82 Úprava hlavného menu



Obr. č. 83 Nové dynamické menu pre orientáciu v systéme

13.11 Úprava URL odkazov na súbory

Zákazník vyžaduje, aby adresa v prehliadači zodpovedala zobrazovanému súboru a aby nemusel kopírovať špeciálny odkaz.

Analýza

URL v prehliadači nie je jednoduché zmeniť. Aj samotné prehliadače zakazujú úpravu URL, aby nedošlo k zneužitiu tejto funkcionality. Riešením je využívanie tzv. fragmentu, ktorý predstavuje v URL časť textu za znakom ‚#' a má určovať pozíciu v rámci dokumentu.

Návrh

Po kliknutí na súbor pridáme do URL fragment s potrebnými identifikačnými údajmi o súbore. Ak sa tento fragment nachádza v odkaze, spracujeme ho a automaticky načítame konkrétny súbor v strome.

Riešenie

Samotná implementácia prebiehala v javascripte zmenou hodnoty window.location.hash. Táto hodnota v sebe nesie fragment a tento fragment nastavíme tak, aby zodpovedal zobrazenému súboru. Pokiaľ sa v URL prehliadača nachádza fragment, overíme či je správny a pokúsime sa načítať takýto súbor, aby sa automaticky zobrazil.

Testovanie

Testovanie prebiehalo manuálne overením funkcionality pridanej v javascripte. Okrem toho bol odstránený starý kód a otestované, či neboli omylom vykonané škody pri odstraňovaní.

13.12 Vytváranie značiek

Možnosť pridávania značiek prostredníctvom file structure obrazovky. Po kliknutí na kód alebo po označení kódu v prehliadači bude implementovaná možnosť vložiť k označenému kódu značku podľa aktuálneho profilu projektu.

Analýza

Zobrazenie menu

Zobrazenie tlačidla by sa malo zobraziť po kliknutí na kód. Avšak je nutné zabezpečiť aj označovanie kódu a zobraziť tlačidlo po označení kódu. Označenie kódu teda vylučuje použitie udalosti „click".

Najlepšie riešenie zobrazenia sa zdá použitie udalosti „mouseup“. Takto sa zobrazí menu aj pri kliknutí ale aj po označení daného kódu.

Poslanie dát o značke

Na poslanie dát je nutné použiť AJAX, ktorý pošle vyplnené údaje o značke metóde na servery, ktorá spracuje údaje a vloží ich do systému ITM. Pred odoslaním samotných dát je nutné získať informácie o aktuálnom profile značiek v danom projekte. Keďže podľa tohto profilu budú ponúknuté značky na výber.

Návrh

Zobrazenie menu

Pred zobrazením menu je nutné vybrať informácie o označenom kóde. V analýze bola vybraná udalosť „mouseup“ je však nutné k nej pridať udalosť „mousedown“ aby bolo možné zaznamenať pozíciu začiatku označovania. Pri týchto dvoch udalostiach je možné vybrať rozmedzie riadkov, na ktorom sa bude budúca značka nachádzať. Označený kód nie je potrebné vyberať. V udalosti „mouseup“ sa zobrazí tlačidlo „pridať značku“.

Po kliknutí na zobrazené tlačidlo je nutné zistiť aktuálny profil projektu to bude musieť byť vykonané AJAX volaním. Metóda na serverovej strane podľa aktuálneho profilu vráti „partial view“, v ktorom bude daný formulár zodpovedajúci profilu.

Poslanie dát o značke

Po vyplnení formuláru programátorom sa pomocou AJAX volanie pošlú vyplnené údaju spolu s rozsahom označeného kódu získaného skorej. Na serverovej strane sa zistí používateľ, ktorý pridal značku a vytvorí sa trieda, ktorá sa vloží do systému ITM.

Riešenie

Zobrazenie menu

Pri implementácií označovania kódu sa vyskytlo niekoľko problémov. Jedným z nich bolo zistenie riadku kódu, na ktorý bolo kliknuté. Keďže kóde je reprezentovaný html tagmi „li“ bolo možné použiť metódu „Index()“. Problém sa naskytol pri klikaní na tieto tagy. Keďže v týchto tagoch sa nachádzaju ďalšie bolo nutné vytvoriť udalosť „mousedown“/“mouseup“ na všetky tieto tagy. Podľa ich pozície sa následne pomocou metódy „parent()“ dalo získať tag „li“ a následne jeho poradie v rámci zoznamu.

Po zistení riadkov bolo nutné spočítať znaky po tieto riadky od začiatku súboru, keďže v systéme ITM sa použil TextualContentSelector, ktorý identifikuje pozíciu značiek podľa indexu znakov. To bolo implementované nasledovne.

```
$("#content ol li").each(function () {
            if ($("#content ol li").index($(this)) < selectedLinesS - 1 ) charA +=
            $(this).text().length - 4;

            if ($("#content ol li").index($(this)) < selectedLinesE - 1 ) charB +=
            $(this).text().length - 4;
})
```

Obr. č. 84 Počítanie znakov od začiatku súboru po dané riadky.

Po spočítaní znakov sa zobrazí tlačidlo „Add Tag“, ktoré vyvolá AJAX. Na serverovej strane sa najprv zistia profily značiek a následne sa v databáze nájde aktuálny profil pre daný projekt. Z profilu je možné vytiahnuť typy značiek, ktoré je možné vložiť a presné polia, ktoré je nutné vyplniť aby bola pridaná. Podľa týchto informácií sa vytvorí a pošle formulár.

Poslanie dát o značke

Po vyplnení informácií vo formulári a odoslaní AJAX. Sa na servery vytvorí trieda:

```
ITMaintenance.Driver.InformationTags.UserInformationTag newTag = new
   ITMaintenance.Driver.InformationTags.UserInformationTag();


newTag.HasBody = new ITMaintenance.Driver.DataTypes.Body() { Type =
   ITMaintenance.Driver.InformationTags.UserInformationTag.RDFTagBody};
```

Obr. č. 85 Vytvorenie inštancie triedy UserInformationTag.

V tejto triede sa vyplnia potrebné informácie, ktoré boli prijaté. Keď sú všetky informácie vložené, vytvorí sa pripojenie na systém ITM a vloží sa daná trieda.

```
ITMServer Connection = ITMServer.CreateFIITConnection();

try // Pridanie tagu
{
 var result = Connection.AddInformationTag(newTag);
 if (result == null)
       {
       throw new Exception("Znacka nepridana" + newTag.ToString());

 }
}
catch (Exception e)
{
       Logger.Error("Exception pri pridávaní značky", e);
       return Json(false);
}

return Json(true);
```

Obr. č. 86 Pripojenie na server a pridanie naplneného tagu.

V prípade nepridaného tagu vráti metóda „AddInformationTag()“ hodnotu „null“ vtedy sa vráti v AJAX volaní návratová hodnota „False“ a na stránke sa zobrazí chybová hláška. V opačnom prípade sa zobrazí pozitívna hláška. A obnoví sa stránka aby bola viditeľná aj novo-pridaná značka.

Testovanie

Testovanie prebiehalo manuálne keďže bolo hlavne nutné otestovať javascript na stránke a správne určenie rozmedzia značky. Tiež bol využívaný firebug na testovanie AJAX volania a správnosti dát.

13.13 SYNCHRÓNNE SCROLLOVANIE PRI ZOBRAZENÍ ZMIEN V SÚBORE

Riešenie

Pridali sme nový javascript „dual-scroll.js“, ktorý zabezpečuje synchrónne posúvanie dvoch susedných kódov v zobrazení histórie.

14 DRUHÝ LÍSTOK

Číslo šprintu: 8

Začiatok šprintu: 27.03.2014

Koniec šprintu: 10.04.2014

Príbehy:

- **Rozsiahle codereview**
- **Zobrazenie konkrétnej notification message v parciálnom view**
- **Zoznam správ má byť globálny**
- **Zmena odkazu na značku v zozname značiek**
- **V značkách zobrazovať meno autora podľa zoznamu aliasov (nie login)**
- **Po filtrovaní zoznamu značiek ostáva v dropdown liste zobrazená rovnaká možnosť**
- **Kontrola typov atribútov pridávaných informačných značiek podľa profilu značky**
- **Podpora pre zobrazenie značiek ukotvených pomocou selektora typu TextualContextSelector**
- **Načítanie zmien v odovzdaní 2541 nikdy neskončí**
- **Pri zobrazení requestov sa zobrazí tabuľka aj keď nie je žiaden request**
- **Zmeniť formát informácie o prebiehajúcom načítaní štatistík**
- **Nastavenie profilu značiek sa vždy inicializuje na "BaseProfile"**
- **Spojiť štatistiky aj pre mená s rôznou veľkosťou písma**
- **Okienko context menu – position**
- **Linky na zvýrazniť (oddeliť od textu) podčiarknutím čiarkovanou (bodkovanou) čiarou**
- **Pri zobrazení zmenených súborov v odovzdaní 2541 dôjde k chybe**
- **Zmena dizajnu domovskej stránky projektov**
- **Zobrazenie informácií o počte výsledkov vyhľadávania v súboroch**
- **Spustenie vyhľadávania pri stlačení klávesy Enter**
- **Nahradenie stránky Under construction domovskou stránkou projektov**
- **Vymeniť radio buttony v tabuľke aliasov za Delete tlačidlá**
- **Menu má privysoký z-index - nastaviť v súlade s ostatnými prvkami**
- **Prerobiť odkaz na správu aliasov, tak aby bol intuitívny**
- **Tlačidlo "Add new notification" v administrácii notifikácií naformátovať ako tlačidlo**
- **Zrušenie tabuľky v tabuľke pre správu používateľov projektu a pre správu repozitárov**
- **Odstránenie čiarkovaného riadku v tabuľke administrátorov**
- **Priame odosielanie zmien v správe používateľov bez použitia Submit tlačidla**
- **V tabuľke pre správu používateľov a** pre správu projektov **projektu nahradiť tlačidlo Select priamym výberom projektu**
- **Tlačidlá pre úpravu, zmazanie a zapnutie emailov v správe notifikácií**

- **DatePicker v štatistikách**
- **Zmena dizajnu tlačidiel**
- **Zobrazenie ikoniek domov a nastavenia vpravo vedľa ikony odhlásenia**
- **Spracovanie notifikácie o vytvorení/zmazaní značky**
- **[BUG] LoadTags.TagsFromFile() - Cast Exception**

14.1 ROZSIAHLE CODEREVIEW

Analýza

Na našom tímovom projekte robí celkovo sedem členov. Každý člen nášho tímu má rôzne skúsenosti s používanými technológiami a s programovaním celkovo. Zdrojový kód, ktorý sme produkovali od začiatku práce na projekte je rôzne kvalitný a postupne sa začínali objavovať rôzne chyby, ktoré častokrát boli spôsobené nepozornosťou, alebo zlou implementáciou, preto bolo nutné ručne prejsť celý zdrojový kód a nájsť čo najväčší počet nedostatkov.

Riešenie

Všetky zdrojové kódy písané v jazyku c# boli súbor po súbore prejdené a skontrolované. Menšie chyby, ako napríklad pridanie atribútov na kontrolu používateľských práv, alebo drobné zmeny v štýle programovania sme opravili. Chyby a nedostatky, ktoré si vyžadovali vyššiu pozornosť (prepísanie metódy, doplnenie funkcionality) sme označili informačnými značkami systému PerConIK. Odhalené veľké chyby sme zaznamenali do systému TFS ako chyby (*Bug*).

14.2 ZOBRAZENIE KONKRÉTNEJ NOTIFICATION MESSAGE V PARCIÁLNOM VIEW

Správu notifikácie je potrebné zobraziť v novom *parciálnom view*, najvhodnejšie vo forme vyskakovacieho okna.

Analýza

Aktuálne sa správa notifikácie zobrazuje na novej stránke. Tento spôsob sa nejaví ako najvhodnejší, preto je potrebné zobraziť správu v novom *parcialnom view*, prípadne iným spôsobom.

Návrh

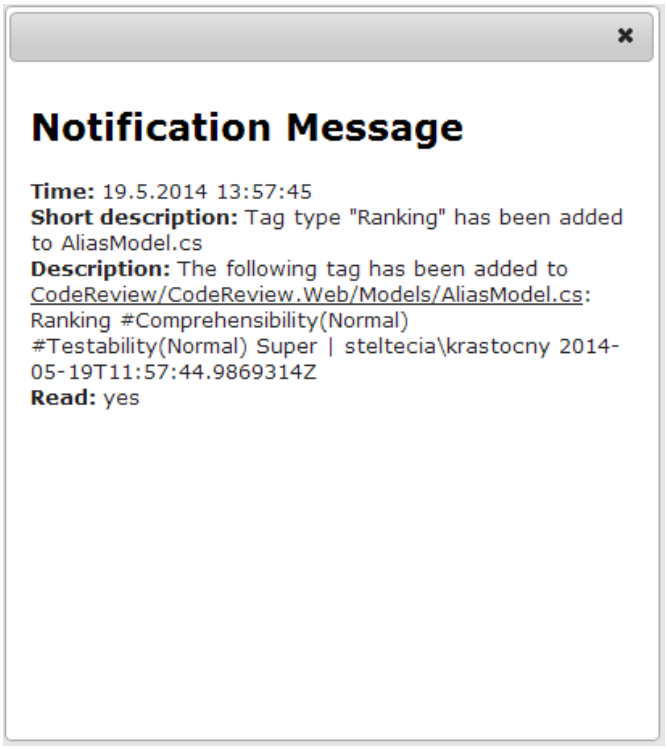
Zobrazenie správy notifikácie urobíme prostredníctvom vyskakovacieho okna – tzv. dialógu (dialógového okna), ktoré bude obsahovať všetky informácie týkajúce sa správy notifikácie, čiže pôvodné *view* s informáciami.

Riešenie

Pre riešenie sme využili možnosti JavaScriptu a jQuery. Pre korektné zobrazovanie vyskakovacieho okna – dialógu bolo potrebné implementovať novú verziu knižnice jQuery.UI, konkrétne verziu 1.10.4.

Po kliknutí na odkaz s notifikáciou prebehne skript, ktorý zobrazí *view* s podrobnosťami notifikácie v dialógovom okne namiesto vytvorenia celej novej stránky a jej zobrazenia.

Pre zobrazenia na rôznych stránkach bolo potrebné k stránkam s rozložením – *layoutom*, pridať na stránky HTML prvok *div*, do ktorého sa malo zobraziť vyskakovacie okno - dialóg. Výsledne zobrazenie správy notifikácie môžeme pozorovať na obrázku č. 87.



Obr. č. 87: Zobrazenie správy notifikácie v dialógu

Testovanie

Testovanie prebehlo vizuálnou kontrolou, pričom sa pozorovalo, či sa dialóg zobrazí správne. V niektorých prípadoch sa však dialóg nezobrazil a informácie o značke sa zobrazili priamo v texte aktuálnej stránky. Tento problém bol pravdepodobne spôsobený nesprávnym umiestnením HTML prvku *div* na stránke, čím sa dialóg nezobrazil korektne v strede, ale na inom mieste ako text na stránke.

14.3 Zoznam správ má byť globálny

Zoznam vypísaných správ notifikácií sa nemá týkať iba jedného vybraného projektu, ale ma zahŕňať všetky projekty.

Analýza

V predchádzajúcom šprinte sme pridali v menu položku so zobrazením správ notifikácií. Tie sme vyfiltrovali na základe používateľa a projektu. Vznikla však nová požiadavka, aby bol zoznam globálny a výpis správ nebol obmedzený na vybraný projekt.

Návrh

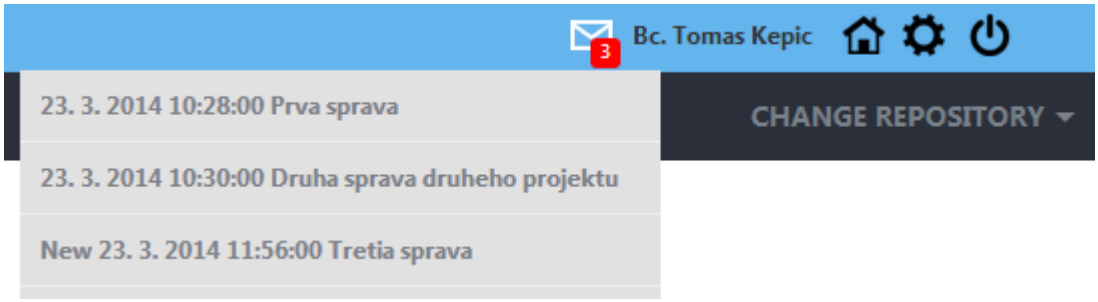
Položku menu výpisu zoznamu správ notifikácií bude potrebne presunúť do iného menu alebo na miesto, ktoré sa netýka iba konkrétneho projektu. Taktiež bude potrebné zrušiť obmedzenia vo funkciách získavajúcich zoznam značiek.

Riešenie

Klikateľné menu zobrazenia značiek sme presunuli na miesto, ktoré je globálne dostupné a nie je potrebné mať vybraný určitý projekt, čiže celkovo do iného *view* v projekte. Vo funkciách ktoré,

získavajú zoznamy značiek sme zrušili obmedzenie na konkrétny projekt. Ostalo obmedzenie iba na konkrétneho používateľa.

Položka so zobrazením správ notifikácií nadobudla na novom mieste nový dizajn, ktorý zapadá do dizajnu okolitých ikon. Taktiež sa zmenil dizajn zobrazenia počtu neprečítaných notifikácií. Nový dizajn môžeme pozorovať na obrázku č. x. Po kliknutí na obálku sa nám zobrazili správy notifikácií zo všetkých projektov, v ktorým je zahrnutý používateľ.



Obr. č. 88: Presunutá položka menu so zobrazením správ notifikácií

Testovanie

Testovanie prebiehalo manuálne s porovnaním v databáze, ako to bolo v prípade keď boli správy na predchádzajúcom mieste v inom menu. Počet neprečítaných notifikácií sa zobrazoval korektne, avšak boli zobrazené pre všetky projekty používateľa, keďže sme zrušili toto obmedzenie.

14.4 ZMENA ODKAZU NA ZNAČKU V ZOZNAME ZNAČIEK

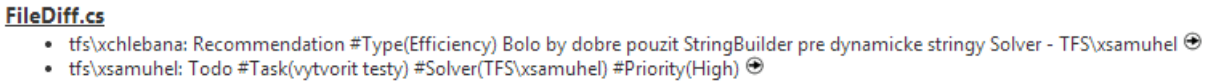
Používateľ nechce pristupovať ku značkám v zozname značiek prostredníctvom odkazu ,,Link“ ale vhodnejším odkazom.

Návrh

Namiesto textového odkazu navrhujeme použiť obrázok, ktorý zobrazuje šípku.

Riešenie

Odkaz ,,Link“ sme nahradili obrázkom šípky. Výsledok môžeme vidieť na obrázku č. 89.



Obr.č. 89: Zobrazenie jedného súboru zo zoznamu značiek s upraveným odkazom na značky

Testovanie

Funkcionalitu odkazu sa nezmenila a zostala zachovaná. Zmenilo sa iba zobrazenie formy odkazu.

14.5 V ZNAČKÁCH ZOBRAZOVAŤ MENO AUTORA PODĽA ZOZNAMU ALIASOV (NIE LOGIN)

Pri zobrazení autora značky je potrebné nahradiť rôzne aliasy rovnakým menom.

Analýza

Značka v systéme má určitého autora. Môže sa stať, že je názov autora rozdielny kvôli rôznym menám v rôznych projektoch alebo systémoch, avšak autor je rovnaký. Na zjednotenie mena autora

slúžia aliasy. V našom systéme existuje spôsob nahradenia aliasov, ktorý je využívaný pri zobrazení štatistík používateľov. Zjednotenie mena autora podľa aliasov je potrebné vykonať aj pri zobrazení autora značky.

Návrh

Preskúmame existujúce riešenie nahradenia aliasov pri štatistikách používateľov a ak to bude možné, aplikujeme ho na nahradenie autorov značiek podľa aliasov.

Riešenie

Existujúce riešenie sme preskúmali, je vhodné a použili sme ho aj pri zobrazení autorov značiek. To funguje na takom princípe, že sa vytvorí zoznam aliasov v systéme, ktorým je priradený autor. Ak meno v značke je rovnaké ako niektorý alias, zobrazíme meno priradené k tomuto aliasu. Ak sa alias nenájde, zobrazíme meno, ktoré je pôvodne uložené v značke.

Testovanie

Nahrádzanie aliasov pri autoroch v značkách sme otestovali. Ak to bolo potrebné a existoval alias, meno autora sa vhodne nahradilo.

14.6 Po filtrovaní zoznamu značiek ostáva v dropdown liste zobrazená rovnaká možnosť

Po vybraní typu značky pre filter pri zozname značiek je potrebné zabezpečiť, aby po jeho aplikovaní zostala vybraná rovnaká možnosť a nenastavila sa ,,defaultná“ hodnota.

Analýza

Momentálne je list typov značiek napĺňaný vo *view*, kde sa vyplnia hodnoty *dropdown listu*. V ňom však nedokážeme vhodne dynamicky nastaviť *selected value*, čiže hodnotu, ktorej zobrazenie ma byť zachované po aplikácii filtra.

Návrh

Namiesto výberu označenej hodnoty pre zobrazenie vo *view* ju vyberieme v *controlleri*, pričom vytvoríme nový *dropdown list* s vyznačenou hodnotou a odovzdáme ho *view*.

Riešenie

Konkrétne možnosti *dropdown listu* vo *view* sme vymazali. Jeho naplnenie je vykonané prostredníctvom premennej, ktorá je poskytnutá *controllerom*. V ňom uvažujeme nad dvoma možnosťami vyplnenia:

1. Prvé zobrazenie zoznamu značiek, kedy je ,,defaultne“ vyznačená možnosť všetkých značiek
2. Výber konkrétneho typu značky a aplikovanie filtra, kedy je vybraná niektorá poskytnutá možnosť

Pri prvom prípade je jednoducho vytvorený *dropdown list* s hodnotami, pričom vybraná je hodnota pre všetky značky. Ak používateľ zobrazí zoznam značiek a následne aplikuje filter, nastane druhá možnosť, kedy v *controlleri* získame označenú hodnotu, ktorú použijeme na označenie zvolenej hodnoty v novovytvorenom *dropdown liste* s rovnakými možnosťami. Hodnoty pre *dropdown list* s vybranou hodnotou sú pomocou *viewbag* poskytnuté *view*, ktoré ich správne zobrazí. Tým zabezpečíme, že v *dropdown liste* ostáva zvolená rovnaká hodnota aj po aplikácii filtra.

Testovanie

Testovanie prebehlo jednoducho vizuálne, kedy sme zvolili typ značky, pričom ten ostal v možnostiach výberu zachovaný a zoznam značiek sa správne prefiltroval.

14.7 KONTROLA TYPOV ATRIBÚTOV PRIDÁVANÝCH INFORMAČNÝCH ZNAČIEK PODĽA PROFILU ZNAČKY

Pri vytváraní značky existujú zadefinované typy, ktoré môže obsahovať značka v danom poli. Je nutné tieto typy kontrolovať, aby nebola vložená zlá hodnota.

Analýza

Validáciu dát podľa typov zadefinovaných v profile je najlepšie robiť už pri zadávaní dát. Preto by bolo dobré implementovať túto validáciu v JavaScripte pri vyplňovaní formulára. Na validáciu rôznych typov dát by mohol byť vhodný regex.

Riešenie

Pri implementácií tejto funkcie musí byť typ poľa prenesený z controllera. Najrýchlejšie riešenie sa zdalo použitie html atribútu: data. Do html atribútu sa uložil typ značky, ktorý sa pri validácií daného poľa použil. V JavaScript funkcií: „ValidateForm()“ bolo implementované overovanie správnosti dát na základe typu značky za pomoci regex výrazu: „`/^[0-9]+$/`“. Keďže zatiaľ bolo nutné rozpoznať iba číselnú hodnotu teda Integer alebo reťazec znakov – string. Boli implementované kontroly len týchto dvoch typov. Typ string bol zadefinovaný ako reťazec znakov, ktorý neobsahuje iba číslice, teda nesedel s regex výrazom: „`/^[0-9]+$/`“.

Funkcia je implementovaná v súbore „add-tags.js“ .

Testovanie

Validácia bola testovaná ručne, boli zadané správne a nesprávne typy do vybraných polí.

14.8 PODPORA PRE ZOBRAZENIE ZNAČIEK UKOTVENÝCH POMOCOU SELEKTORA TYPU TEXTUALCONTEXTSELECTOR

Analýza

Zobrazovanie značiek sa uskutočňuje pomocou dvoch typov selectorov. Prvý typ, ktorý zobrazuje značky na daných riadkov bol implementovaný skôr. Druhým typom je TextualContextSelector, ktorý zobrazuje značky pomocou indexu znakov. Značky tohto typu sú pridávané práve vo webovom rozhraní.

Pri zobrazení značky je nutné označiť kód, od zadefinovaného začiatočného znaku po koncový znak značky. Sčasti využiť je možné existujúce riešenie pre zobrazovanie značiek podľa riadkov.

Návrh

Pre zobrazovanie značiek podľa riadkov je pridávaný html tag ako označenie začiatočného a koncového riadku značky. Keďže označovanie po presných znakoch nie je až také potrebné najvýhodnejším spôsobom sa ukazuje „zaoukrúhliť“ znaky na riadky pri zobrazovaní. Čiže ak znak podľa zadefinovaných indexov zasahuje do riadku označí sa celý tento riadok. Preto je nutné vytvoriť prepočítavanie znakov na riadky. A potom klasicky zobraziť podľa existujúceho riešenia.

Riešenie

Existujúce riešenie je pridávanie html tagov v príslušnom view. Do toho view bola pridaná podmienka pre TextuelContextSelector.

```
if (Model[x].SelectorType == "itm:TextualContextSelector")
   {

          if (charCountA > Model[x].StartIndex && charCountB <= Model[x].StartIndex &&
              charCountA > Model[x].EndIndex && charCountB <= Model[x].EndIndex)
             {
                 @:tagstart@(x) tagend@(x)
                 continue;
             }
          if (charCountA > Model[x].StartIndex && charCountB <= Model[x].StartIndex)
             {
                 @:tagstart@(x)
             }

          if (charCountA > Model[x].EndIndex && charCountB <= Model[x].EndIndex)
             {
                 @:tagend@(x)
             }

  }
```

Obr. č. 90 Vkladanie html tagu s id podľa čísla a začiatku alebo konca zobrazovanej značky.

Prvý prípad v Obr. č. 90 je, že značka je zobrazená iba na jednom riadku. Takto sa iteruje po každom riadku v kóde a podľa údajov sa označia dané riadky.

Testovanie

Testovanie bolo uskutočňované spolu s testovaním pridávania značiek, keďže pri pridávaní značiek cez webové rozhranie sa využíva práve tento typ značiek.

14.9 Pri zobrazení requestov sa zobrazí tabuľka aj keď nie je žiaden request

Zákazník si neželá, aby sa zobrazovala prázdna tabuľka, ak neexistujú žiadne požiadavky v systéme.

Riešenie

Riešenie tohto problému je veľmi triviálne. Stačí, aby sme pridali podmienku, že tabuľka sa má vykresliť iba ak je počet požiadaviek viac ako 0, inak vypíšeme informáciu o stave požiadaviek.

Testovanie

Testovanie prebehlo manuálne overením výpisu pri rôznych stavoch požiadaviek.

14.10 Zmeniť formát informácie o prebiehajúcom načítaní štatistík

Zákazník si želá, aby výpis stavu o prebiehajúcom načítavaní štatistík vyzeral inak.

Riešenie

Riešenie bolo v tomto prípade triviálne. Stačilo vymeniť animovaný obrázok znázorňujúci načítavanie za bledý variant. Výsledkom je, že obrázok je vidno aj na tmavom podklade.

Testovanie

Testovanie prebehlo vizuálnym overením zmeny obrázka.

14.11 Nastavenie profilu značiek sa vždy inicializuje na "BaseProfile"

Riešenie

Do kódu bola pridaná podmienka pre zobrazovanie práve zvoleného profilu z databázy.

14.12 Spojiť štatistiky aj pre mená s rôznou veľkosťou písma

Riešenie

Pridaná podmienka, pre porovnanie mien aj s rôznou veľkosťou znakov.

14.13 Okienko context menu – position

Riešenie

Chyba zobrazenia bola upravená úpravou štýlou v CSS – pridaný *z-index*.

14.14 Linky na zvýrazniť (oddeliť od textu) podčiarknutím čiarkovanou (bodkovanou) čiarou

Riešenie

Úprava štýlov pre linky (<a>) bola vykonávaná pridaním štýlu *text-decoration*.

14.15 Pri zobrazení zmenených súborov v odovzdaní 2541 dôjde k chybe

Chyba nastane pri prechode na nasledujúcu adresu:

https://147.175.149.11/Codereview/CodeChanges/SelectedChangesetChanges?changesetId=2541&projectId=2&repositoryId=1

Riešenie

Chybu spôsoboval pokus o načítanie súboru s nepodporovaným formátom. Riešením bolo to, že sme pridali filter na súbory s nepodporovaným formátom. Pri pokuse o zobrazenie týchto súborov sa zobrazí informácia, že typ súboru nebolo možné zobraziť.

14.16 Zmena dizajnu domovskej stránky projektov

(Product Backlog Item 873): Úlohou bola vymyslieť a navrhnúť novú domovskú schránku pre projektov.

Riešenie

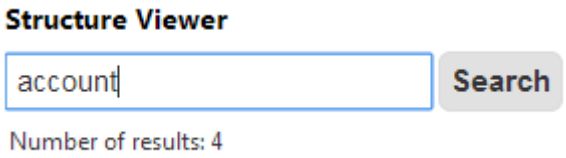
V rámci šprintu bola vymyslená dizajn novej schránky. Nová domovská stránka bude obsahovať tak isto možnosť vytvorenia nového logického projektu ako doteraz a budú zobrazené aj projekty používateľa. Tieto prvky budú ponechané, zmena bude iba v ich zobrazeniu, aby celkový dizajn stránky bol príjemný. Ďalej doteraz nevyužité miesto na ľavej strane stránky bude obsahovať informácie o projekte ako motiváciu a ciele projektu a o našom tíme. V rámci šprintu bol vytvorený iba návrh stránky. Implementácia bola presunutá do ďalšieho šprintu

14.17 Zobrazenie informácií o počte výsledkov vyhľadávania v súboroch

(Product Backlog Item 872):

Riešenie

Do funkcie vyhľadávania nad menami zdrojových súborov a kódových entít bol pridaný počet výsledkov vyhľadávania, ktorá bola implementovaná ako *JavaScript* metóda. Pridaná funkcionalita je znázornená na obrázku č. 91.



Obr. č. 91 Počet výsledkov vyhľadávania

14.18 Spustenie vyhľadávania pri stlačení klávesy Enter

(Product Backlog Item 871):

Riešenie

Bola pridaná možnosť spustenia vyhľadávania nad menami zdrojových súborov a kódových entít aj stlačením klávesy „*Enter*“.

14.19 Nahradenie stránky Under construction domovskou stránkou projektov

(Product Backlog Item 870):

Riešenie

Príbeh bol posunutý do ďalšieho šprintu, lebo v aktuálnom vznikla iba návrh novej domovskej stránky.

14.20 Vymeniť radio buttony v tabuľke aliasov za Delete tlačidlá

(Bug 848):

Riešenie

Zmena bola vykonaná vo *View* pre správu aliasov. Nová tabuľka je na obrázku č. 92. Namiesto radio button-ov každý riadok tabuľky obsahuje tlačidlo „*Delete*“, ktoré zavolá *HttpPost* volanie na vymazanie aliasu používateľa z databázy.

Aliases of test

Alias Name	Server	Select
test1	http://mirawen.fiit.stuba.sk:8080/tfs/FIIT	Delete
test1	http://tfs.fiit.stuba.sk:8080/tfs/StudentsProjects	Delete
test	http://mirawen.fiit.stuba.sk:8080/tfs/FIIT	Delete
test	http://mirawen.fiit.stuba.sk:8080/tfs/TeamProjects	Delete

Obr. č. 92 Vymazanie aliasov

14.21 Menu má privysoký z-index - nastaviť v súlade s ostatnými prvkami

Riešenie

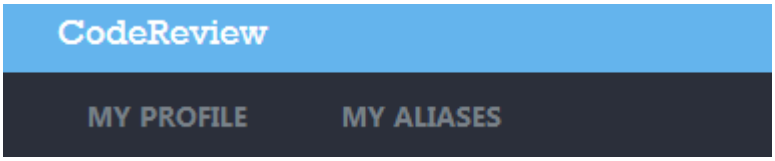
Štýl menu bol upravený zmenou *z-index*-u prekrývajúcich sa prvkov.

14.22 Prerobiť odkaz na správu aliasov, tak aby bol intuitívny

(Product Backlog Item 846):

Riešenie

Nový odkaz na správu aliasov bol pridaný do menu, pre správu používateľského konta. Tento nové menu bol vytvorený rovnakým dizajnom ako má menu pre správu projektov. Nový odkaz je na obrázku č. 93.



Obr. č. 93 Nový odkaz na správu aliasov

14.23 Tlačidlo "Add new notification" v administrácii notifikácií naformátovať ako tlačidlo

Pre zvýšenie prehľadnosti sa požaduje aby nebolo "Add notification" zobrazené ako klikateľný text ale ako tlačidlo.

Čísla changesetov: 1186

View:

CodeReview.Web.Views.Notification.ManageNotification

14.24 Zrušenie tabuľky v tabuľke pre správu používateľov projektu a pre správu repozitárov

Zákazník požaduje aby tabuľka správy používateľov projektu a repozitárov bola zobrazená inak, a to bez dvojitej čiary v orámovaní.

Riešenie

Čísla changesetov: 1187

View:

CodeReview.Web.Views.ProjectManager.Swapper

CodeReview.Web.Views.ProjectManager.SwapperUser

14.25 Odstránenie čiarkovaného riadku v tabuľke administrátorov

Zákazník požaduje aby bol odstránený prebytočný riadok v tabuľke pre správu administrátorov.

Návrh

Odstránenie prebytočného riadku a zmena úpravy tabuľky, pridanie hlavičky do tabuľky.

Riešenie

Čísla changesetov: 1187

View:

CodeReview.Web.Views.Users.AdminManager

14.26 Priame odosielanie zmien v správe používateľov bez použitia Submit tlačidla

Zákazník požaduje aby bolo používateľovi umožnené zadávať pri správe používateľov zmeny bez nutnosti použitia submit tlačidla.

Analýza

Rovnako ako v časti 11.3 s tým rozdielom, že používateľ už nemusí svoj výber potvrdzovať.

Riešenie

Čísla changesetov: 1190

Controller:

CodeReview.Web.Controllers.UsersController

metódy:

public ActionResult ProjectRolesManager(int projectId)

public ActionResult ChangeUserRoles(int projectId, int userId, int role)

Model:

CodeReview.Web.Models.UserRoleInProject

View:

CodeReview.Web.Views.Users. ProjectRolesManager

CodeReview.Web.Views.Users.EditorTemplates.UserRoleInProject

Testovanie

Vstup: Každému používateľovi bola zvolená iná rola, ako mu bola pôvodne priradená. Potvrdenie

Výstup: Po každej zmene kontrola v databáze.

14.27 V TABUĽKE PRE SPRÁVU POUŽÍVATEĽOV PROJEKTU A PRE SPRÁVU PROJEKTOV NAHRADIŤ TLAČIDLO SELECT PRIAMYM VÝBEROM PROJEKTU

Zákazník požaduje aby bolo používateľovi umožnené zadávať pri správe používateľov projektu a pri správe projektov zmeny bez nutnosti použitia Select tlačidla.

Riešenie

Čísla changesetov: 1190

Controller:

CodeReview.Web.Controllers.ProjectsManagerController

metódy:

[HttpGet]

public ActionResult Swapper()

public ActionResult SwapperUser()

[HttpPost]

public ActionResult Swapper(SwapperModel model)

public ActionResult SwapperUsers(SwapperModel model)

Model:

CodeReview.Web.Models.SwapperModel

View:

CodeReview.Web.Views.ProjectManager.Swapper

Testovanie

Vstup: Voľba rôznych projektov.

Výstup: Sledovanie zmien v rozhraní a kontrola s databázou.

14.28 TLAČIDLÁ PRE ÚPRAVU, ZMAZANIE A ZAPNUTIE EMAILOV V SPRÁVE NOTIFIKÁCIÍ

Do správy notifikácií je pre zvýšenie prehľadnosti pre používateľa potrebné pridať ikony, ktoré budú symbolizovať úpravu, zmazanie a zapnutie emailov.

Riešenie

Čísla changesetov: 1186

View:

CodeReview.Web.Views.Notification.ManageNotification

CodeReview.Web.Views.Notification.EditorTemplates.NotificationModel

Controller:

CodeReview.Web.Controllers.NotificationController

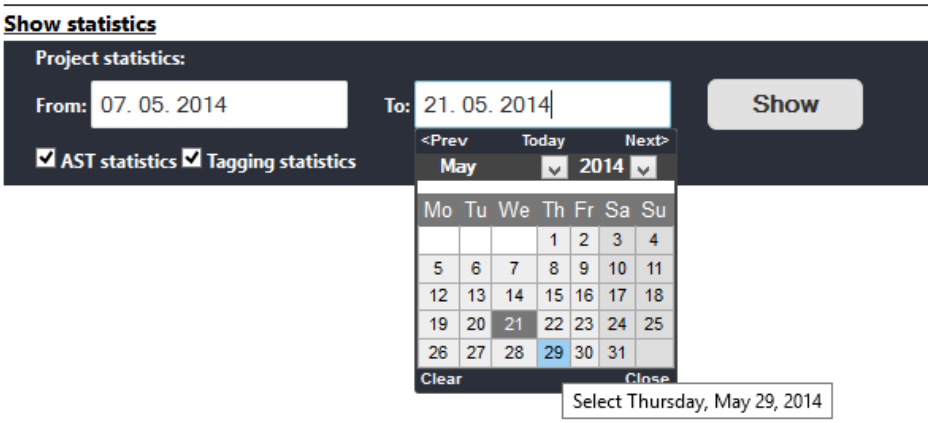
14.29 DATEPICKER V ŠTATISTIKÁCH

Pre zjednodušenie zadávania dátumov pri zobrazovaní štatistík sme sa rozhodli do systému pridať DatePicker.

Návrh

Bolo potrebné vyhľadať vhodný JavaScript vytvárajúci kalendár. Tento kalendár bol následne upravený aby dizajnovo zodpovedal nášmu systému. Zvolili sme preddefinované dátumy na zobrazenie štatistík z posledných dvoch týždňov.

Riešenie

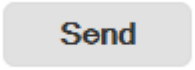


Obr. č. 94 DatePicker

14.30 ZMENA DIZAJNU TLAČIDIEL

Riešenie

Upravili sme tlačidlá v našom systéme keďže dizajnovo nezodpovedali ostatným prvkom.



Obr. č. xx Nový dizajn tlačidiel

14.31 Zobrazenie ikoniek domov a nastavenia vpravo vedľa ikony odhlásenia

Riešenie

Obr. č. 95 Ikony presunuté do pravého horného rohu

14.32 Spracovanie notifikácie o vytvorení/zmazaní značky

Zákazník si želá úplné spracovanie notifikačnej správy v presne stanovenom formáte.

Analýza

V tejto úlohe bude potrebné zistiť, aký je presný formát prijatej odpovede, a čo táto odpoveď v sebe obsahuje. Zvyšné potrebné údaje bude potrebné získať inak.

Návrh

Spracovanie bude prebiehať v nasledujúcich krokoch:

1. Prijatie odpovede a jej deserializácia do objektu EventListenerResponse.
2. Získanie konkrétnej značky a údajov z nej.
3. Zistenie potrebných informácii o súbore.
4. Formátovanie notifikačnej správy.

Riešenie

Samotné riešenie je potom vytvorené na základe návrhu.

1. Najprv prijmeme odpoveď vo forme EventListenerResponse.
2. Táto odpoveď obsahuje InformationTag, ktorý prekonvertujeme na UserInformationTag.
3. UserInformationtag v sebe obsahuje TargetElement, v ktorom sa nachádza URI dokumentu. V tomto URI sa nachádza aj VersionId súboru.
4. Na základe získaného VersionId zavoláme AST-RCS službu, ktorá nám pošle údaje o tomto súbore.
5. Údaje spracujeme a vytvoríme notifikačnú správu.
6. Kompletnú notifikačnú správu označíme za neprečítanú a vložíme ju do databázy.

Testovanie

Bohužiaľ testovanie tohto komponentu je veľmi náročné a to hneď z niekoľkých dôvodov.

1. Služba, ktorá posiela správu do nášho systému sa nedá manuálne zavolať.
2. Iba produkčný server dokáže prijímať správy. Testovanie nie je možné v práve vyvíjanej verzii.

Vďaka týmto dôvodom je testovanie tejto služby veľmi náročné (takmer nemožné) a samotné testovanie bude predmetom diskusie na ďalších stretnutiach.

14.33 [BUG] LOADTAGS.TAGSFROMFILE() - CAST EXCEPTION

V systéme nastáva chyba pri pretypovaní *selectoru* značky a systém tak padne pri získavaní značiek z určitého súboru skôr, ako by do *selectoru* priradil hodnotu *null*.

Návrh

Chybu navrhujeme ošetriť zalogovaním problému.

Riešenie

V prípade, že *selector* značky nie je možné pretypovať, čo je však prípad, ktorý nenastáva často, tak tento problém zaznamenáme v logu a do *selectoru* vložíme hodnotu *null*.

Testovanie

Pri testovaní problematického súboru už systém nepadol a chyba sa zalogovala v externom súbore.

15 TRETÍ LÍSTOK

Číslo šprintu: 9

Začiatok šprintu: 10.04.2014

Koniec šprintu: 28.04.2014

Príbehy:

- **Na začiatku zdrojového súboru je zobrazený riadok navyše**
- **Všetky podstránky majú title „Welcome to CodeReview web“**
- **Zmeniť JS zvýrazňujúci kód**
- **Dialógy pre pridanie značky a zobrazenie notifikácie zjednotiť s použitím Dialog-u z jQuery**
- **Zjednotenie výšky stromu a zdrojového kódu**
- **Zmazanie značiek so selektorom iným ako LineSelector**
- **Opravenie synchrónneho scrollovania**
- **Zmazanie CR projektu**
- **Master alias používateľov**
- **Farebné zvýraznenie autora odovzdania v histórii a filtrovanie histórie podľa autorov**
- **Úprava dizajnu Add Tag**
- **Pri zmene veľkosti okna sa pokazí dizajn štatistík**
- **Filtrovanie zoznamu značiek podľa typov a autorov**
- **Automatická aktualizácia počtu neprijatých notifikácií**
- **Zobrazenie "loaderu" pri načítaní zoznamu značiek**
- **Filter CR a AST projektov v správe repozitárov a filter projektov a používateľov v správe používateľov projektu**
- **Zobraziť mená používateľov v správe práv používateľov projektu**
- **Usporiadanie zoznamov podľa abecedy**
- **Zmena dizajnu domovskej stránky projektov**
- **Nahradenie stránky Under construction domovskou stránkou projektov**
- **Context menu v strome je prekryté kódom**
- **Ošetrenie pridania prázdneho aliasu (prázdny string)**
- **[BUG] Značky sa nezobrazujú korektne**

15.1 NA ZAČIATKU ZDROJOVÉHO SÚBORU JE ZOBRAZENÝ RIADOK NAVYŠE

(Bug 1004):

Riešenie

Tento *bug* vznikol pri zmene zobrazenia značiek. Chyba bola odstránená pomocou drobných zmien vo *View FileRequestPatial*.

15.2 VŠETKY PODSTRÁNKY MAJÚ TITLE "WELCOME TO CODEREVIEW WEB"

(Bug 1030): Úlohou bola zabezpečiť správne zobrazenie *Title* jednotlivých stránok.

Riešenie

Pre každú stránku bol pridaný správny názov danej stránky. Zobrazenie *Title* stránky bolo definované v *Layout*-e pre stránky.

15.3 ZMENIŤ JS ZVÝRAZŇUJÚCI KÓD

(Product Backlog Item 1005): Úlohou je vykonať zmenu v *JavaScript-e*, ktoré slúži na zvýraznenie zdrojových kódov, ktorý spôsobil problémy s novou funkcionalitou pre pridávanie značiek.

Návrh

Zvýrazňovač *google-pretify* štandardne ofarbuje aj pozadie parných riadkov. Pri prechode kurzorom nad riadkami bolo prefarbené pozadie riadkov. Na vyriešenie tejto chyby je potrebné preskúmať možnosti zvýrazňovača.

Riešenie

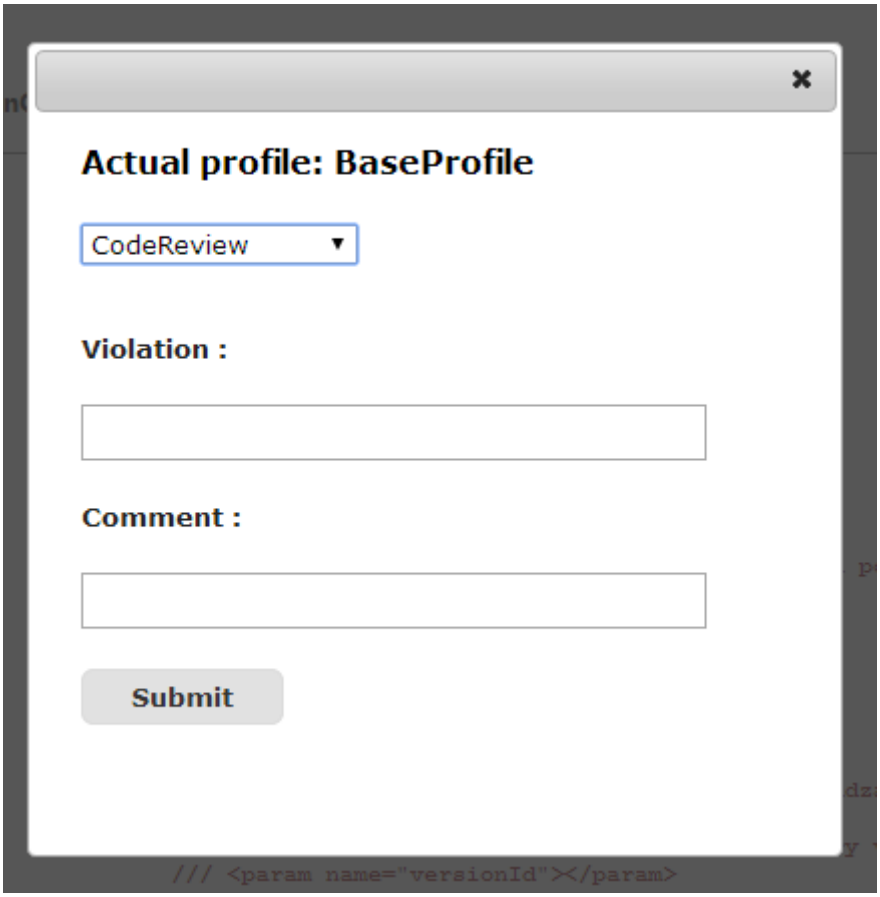
Boli vykonané zmeny v *prettify.css*, ktorý obsahuje dizajnové nastavenie zvýrazňovača. Pozadie každého riadku bolo nastavené na bielu farbu, vďaka čomu bol problém vyriešený.

15.4 DIALÓGY PRE PRIDANIE ZNAČKY A ZOBRAZENIE NOTIFIKÁCIE ZJEDNOTIŤ S POUŽITÍM DIALOGU Z JQUERY

(Product Backlog Item 1031): Pre zobrazenie notifikácií bol použitý *jQuery Dialog*. Pre dodržanie konzistentného dizajnu stránky bolo potrebné zjednotiť dizajn dialóg pre pridávanie značky a dialóg notifikácie.

Riešenie

Rozhodli sme sa použiť *jQuery Dialog* pre dialóg pridávania značiek, ktorý sa nachádza na obrázku č. 96.



Obr. č. 96 Nový dialóg pre pridávanie značky

15.5 Zjednotenie výšky stromu a zdrojového kódu

Zákazník požaduje, aby zobrazený zdrojový kód mal rovnakú výšku akú má aj zobrazená stromová štruktúra.

Riešenie

Samotné riešenie je pomerne jednoduché, ale napriek tomu si vyžaduje rozsiahlejšie úpravy v HTML dokumente ako aj v dokumente CSS. Element, ktorý v sebe zobrazuje stromovú štruktúru aj element, ktorý v sebe zobrazuje zdrojový kód budú mať pevne nastavenú výšku 700 pixelov. Ak tieto údaje vyžadujú viac priestoru, samotný CSS definuje túto plochu ako scrollovateľnú.

Testovanie

Testovanie prebiehalo manuálne a vizuálne overením zobrazenia stromovej štruktúry a rozsiahleho zdrojového kódu. Samotný kód je precízne zarovnaný s elementom, ktorý zobrazuje stromovú štruktúru. Zobrazenie je presnejšie a úhľadnejšie.

15.6 Zmazanie značiek so selektorom iným ako LineSelector

Zákazník požaduje, aby bolo možné mazať značky priamo cez naše webové rozhranie. Značky, ktoré sú vložené priamo v kóde mazať možné nebude.

Analýza

Pri tejto úlohe musíme zistiť, akým spôsobom je možné mazať značku zo systému ITM. Následne musíme implementovať funkcionalitu tak, aby sa značka dala jednoducho odstrániť, ideálne bez toho aby musel používateľ obnoviť stránku v prehliadači.

Návrh

Samotný návrh riešenia sa skladá z dvoch hlavných častí:

- Služba na mazanie značiek, ktorá ako POST parameter prijíma URI značky
- Javascriptový kód na asynchrónne volanie služby

Potrebné taktiež bude pridať klikateľný odkaz ku každej značke, ktorý túto akciu vyvolá.

Riešenie

Samotné riešenie bolo na základe návrhu pomerne jednoduché. Bola vytvorená „action“, ktorá ako POST parameter prijíma URI značky. Tento parameter je potom spracovaný a následne je zavolaná služba ITM servera na mazanie značky.

Bola taktiež pridaný javascriptový kód, ktorý asynchrónne volá túto službu bez toho aby bolo potrebné obnoviť stránku v prehliadači. Po zmazaní značky je samotná značka zmazaná dynamicky aj s HTML dokumentu.

Testovanie

Testovanie prebiehalo skúšaním funkcionality a overovaním či značka reálne bola odstránená z dokumentu. Pri tejto úlohe nenastali problémy a všetko fungovalo tak ako malo.

15.7 Opravenie synchrónneho scrollovania

Zákazník požaduje, aby fungovalo synchrónne scrollovanie vo všetkých častiach webového rozhrania.

Riešenie

Problém bol spôsobený pridávaním scrollovateľného obsahu až po základnom načítaní stránky pomocou javascriptu. Problém bol odstránený aplikovaním synchrónneho scrollovania až po tom, ako sa takýto obsah načítal do HTML dokumentu.

Testovanie

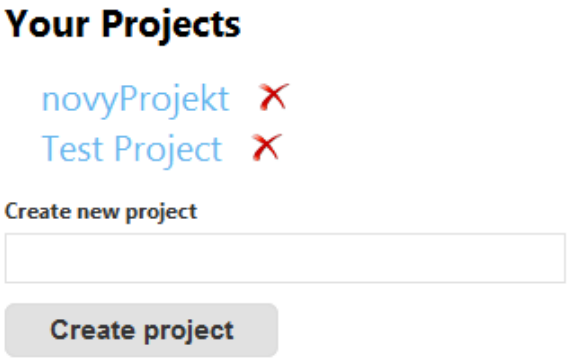
Testovanie bolo veľmi jednoduché a prebiehalo manuálne testovaním synchrónneho scrollovania v prehliadači. Testovanie bolo úspešné.

15.8 Zmazanie CR projektu

Nedovoliť zmazať projekt ak má ešte aj iného administrátora a upraviť selecty na projekty, tak aby bol zvažovaný Deleted flag.

Riešenie

Súčasný databázový model nepodporoval označenie CR projektu ako zmazaného, preto bolo potrebné toto označenie pridať do databázového modelu. Pre vymazávanie CR projektu bolo potrebné vytvoriť formulár pre používateľa, aby vedel ľahko zmazať požadovaný projekt cez používateľské rozhranie.



Obr. č. 97 Tlačilo pre zmazanie CR projektu

15.9 Master alias používateľov

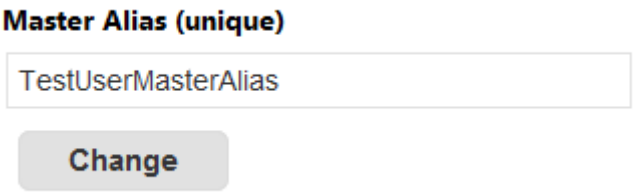
Master alias je unikátny alias, ktorý slúži na jedinečnú identifikáciu používateľa v našom systéme.

Analýza

Pre pridanie Master Aliasu je potrebné upraviť databázový model, na podporu master aliasu, pridanie formulára, pomocou ktorého sa bude tento Master Alias dať upraviť. Následne bude potrebné zmeniť aj niektoré funkcie systému, ktoré doteraz používali Alias, alebo meno používateľa.

Riešenie

Databázový model doteraz nepodporoval ukladanie MasterAliasu používateľa, preto bolo nutné model upraviť. Do tabuľky používateľ sme pridali nový atribút „MasterAlias“. Tento atribút je unikátny v rámci celého systému. Pre existujúcich používateľov sme skopírovali prihlasovacie meno používateľa, ktoré je tiež unikátne, ale nie je ho možné v systéme zmeniť. Na nasledujúcom obrázku je zobrazený formulár na zmenu Master Aliasu.



Obr. č. 98 Formulár pre zmenu Master Aliasu

15.10 Farebné zvýraznenie autora odovzdania v histórii a filtrovanie histórie podľa autorov

Analýza

Filtrovanie a farebné zobrazenie autorov odovzdaní kódov je nutné implementovať v grafe histórie projektu, tak aby bolo vizuálne ľahko rozlíšiteľné, ktoré odovzdanie patrí, ktorému autorovi. Preto je dobré prideliť farbu každému autorovi v rámci grafu ale aj v popise daného odovzdania. Graf je vykresľovaný za pomoci HTML5 canvas. Preto zmena farby grafu by nemala byť náročná.

Je nutné vytvoriť filter, ktorý bude zobrazovať všetkých programátorov daného projektu a k nemu prislúchajúcu farbu.

Návrh

Pri načítaní grafu istého projektu by sa mali načítať všetky odovzdania kódov. Teda zo všetkými programátormi. Filter by bolo možné implementovať ako množina check boxov na stránke. AJAX volaním by sa odoslali vyfiltrované mená a zobrazil by sa upravený graf.

Farby jednotlivých programátorov by mali byť pre lepší výkon zadefinované staticky v dostatočnom množstve, ktoré sa budú prideľovať jednotlivým programátorom pri vytváraní view. Najlepšie riešenie pre uloženie farieb jednotlivých programátorov sa zdá byť uloženie kódu do data atribútu html tagu. Pri zobrazení jednotlivých popisov sa potom iba vyberie farba z html tagu daného programátora.

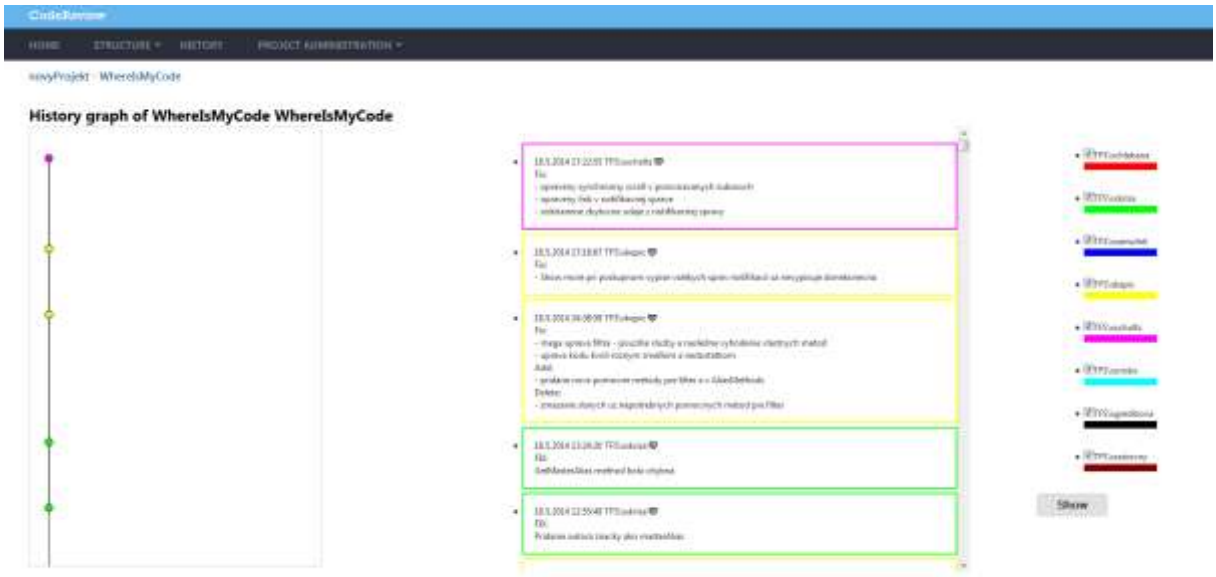
Riešenie

Pri filtrovaní autorov sa odošle AJAX volanie, ktoré zavolá controller: ranchGraph/EntityBranchGraph. Tu sa spracujú informácie a vyfiltrujú odovzdania kódu podľa prijatých autorov. Vo view sa zadefinuje každému autorovi farba, ktorou sa neskôr v JavaScripte zafarbia jednotlivé odovzdania.

```
    var users = new Array();
    $("#userList li div").each(function () {
        var color = $(this).parent().data("color");
        $(this).css({ "background-color": color });
        users.push($(this).data("userid"));
    });
    for (var i = 0; i < users.length; i++) {
        $("." + users[i]).each(function () {
            $(this).css({ "border-color": $("#" + users[i]).parent().data("color")});
        });
```

Obr. č. 99 Zafarbenie odovzdaní podľa autorov.

Táto farba je definovaná pri liste autorov a podľa nej sa zafarbia aj ostatné položky podľa classy jednotlivých tagov.



Obr. č. 100 Ukážka implementovaného zafarbovania grafu podľa autorov.

15.11 Úprava dizajnu Add Tag

Tlačidlo „Add tag“ bolo potrebné upraviť pre sprehľadnenie a zosúladenie z dizajnom systému.

Návrh

Umiestnenie tlačidla sme určili na pozíciu nad zvoleným riadkom do ktorého chceme pridať značku. Tlačidlo má byť priehľadné kým kurzor myši nebude nad tlačidlom. Pri zmene pozície v kóde sa tlačidlo skryje.

Riešenie

Upravili sme Javascript určujúci pozíciu tlačidla „Add tag“. Upravili sme štýlu pre tlačidlo tak aby bol jeho dizajn zosúladený s dizajnom systému. Doplnený javascript pre skrytie tlačidla pri zmene pozície (scroll) v zdrojovom kóde.

Obr. č. 101 Zobrazenie tlačidla „Add tag“

15.12 Pri zmene veľkosti okna sa pokazí dizajn štatistík

Riešenie

Štýly okna so štatistikami boli zle určené. Úpravou štýlov na pevnú šírku okna sme problém vyriešili.

15.13 Filtrovanie zoznamu značiek podľa typov a autorov

Používateľ chce okrem filtra typov značiek aplikovať aj filter autorov značky.

Analýza

V aktuálnom riešení je implementovaný filter typov značiek. Ten je vykonaný postupným prehľadávaním súborového systému repozitáru a zisťovaním, aké značky súbory obsahujú. Nájdené značky sa zaznamenávajú do zoznamu pozostávajúceho z nášho modelu, ktorý obsahuje meno súboru, ID súboru a zoznam značiek v súbore. Tieto získané údaje sú následne prefiltrované a zobrazené.

Pre aplikáciu filtra však existuje ešte druhá možnosť, ktorú sme uviedli pri možnosti získania zoznamu značiek. Táto možnosť pozostáva z volania služby, ktorej poskytneme zoznam Id súborov a projektu (resp. repozitáru) a vráti nám zoznam značiek. Dôležitou a zaujímavou funkcionalitou je však to, že do tejto služby je možné vložiť obmedzenie na typ značky alebo autora. Takto by sme nemuseli používať naše rôzne funkcie na prehľadávanie súborov a aplikovanie filtrov, ale stačilo by zavolať službu. Rozdielom vo výstupe je však to, že funkcia vracia iba zoznam značiek, my sme však naplnili náš vytvorený model. Získané značky však obsahujú údaj o tom, v akom súbore alebo entite sa nachádzajú, preto by bolo možné zistiť tieto údaje dodatočne, spracovať ich a naplniť náš model. Tieto údaje sú dôležité preto, aby sme mohli vytvoriť odkaz na značku v konkrétnom súbore.

Návrh

Pri aplikovaní dvoch filtrov neurobíme rozšírenie nášho doterajšieho riešenia manuálneho zisťovania značiek, ale implementujeme volanie služby, pričom získané údaje upravíme na náš model, ktorý bude jednoduché zobraziť tak, ako je to teraz implementované.

Riešenie

Keďže sme sa rozhodli volať službu, musíme získať potrebné vstupne údaje. Id projektu, resp. repozitáru poznáme. Zoznam Id súborov získame prebehnutím súborovej štruktúry projektu, pričom nebudeme zisťovať značky v súboroch, ale zapamätáme si iba Id súborov.

Údaje pre obmedzenie hľadania získame z filtra podľa voľby používateľa. Typy značiek sme už však nevložili do *dropdown listu* pomocou *controllera* manuálne natvrdo, ale získali sme typy podľa zvoleného značkovacieho profilu k projektu a získanými hodnotami sme naplnili *dropdown list*. Typ značky pri volaní služby je rovnaký text ako v naplnených možnostiach, iba sa pridá predpona „itmut:“.

Pri získaní vybraného autora je postup iný. Vo voľbách v *dropdown liste* sme zobrazili údaj *DisplayName* z databázy, čiže mená používateľov pre zobrazenie podľa vybraného projektu. Každá položka má svoje Id, ktoré je zhodné s Id používateľom v databáze. Do služby pre získanie značiek však nevkladáme *DisplayName* používateľa ani ID, ale jeho *MasterAlias*, ktorý sme zistili pomocou poskytnutej získanej hodnoty Id vo vybranej položke.

Poskytnutím potrebných údajov sme vykonali volanie služby, ktorá nám vrátila zoznam značiek. Ten je však potrebné spracovať vo viacerých krokov.

Prvým je spracovanie značiek a ich úprava na model značiek, s ktorým pracujeme v systéme, čiže z množstva informácií o značke vyberieme iba potrebné údaje pre ďalšiu prácu. Pri tomto spracovávaní taktiež upravíme meno autora značky. Jednu úpravu sme už implementovali, kedy sme preskúmali rôzne aliasy a nahradili ich rovnakým menom. Toto meno teraz nahradíme *MasterAliasom*. Ak sa k nemu však neviem dopracovať vzhľadom na nedostatok údajov v databáze, necháme v značke pôvodné meno.

Po prvotnom spracovaní značiek je potrebné vykonať úpravu na ďalší typ formátu, ktorý je poskytnutý pre zobrazenie vo *view*. Je ním zoznam vytvorený z modelu, ktorý obsahuje názov súboru, Id súboru a príslušné značky. Pri značkách však nemáme priamo určené Id súboru a jeho názov. Značka však obsahuje údaj *Target Element*, ktorý obsahuje adresu v ktorej je údaj buď o Id kódovej entity alebo súboru, v ktorom sa nachádza daná značka. Túto adresu ,,rozparsujeme“, zistíme či ide o Id súboru alebo kódovej entity, čo sa dá vyčítať z danej adresy, a na základe volania služby zistíme údaje o súbore, prípadne najprv o entite a potom o súbore v ktorom sa nachádza. Zo získaných údajov o súbore vieme následne určiť meno súboru, v ktorom sa nachádza značka. Ku každej značke týmto spôsobom zistíme Id súboru a názov, v ktorom sa nachádza. Pritom si vedieme zoznam Id súborov s menami, aby sme nemuseli stále volať služby na získanie súboru v prípade, ak súbor obsahuje viacero značiek a už sme určili jeho meno.

Získali sme teda zoznam značiek, ktorý obsahuje aj údaj o mene a Id súboru. Pre našu potrebu a ďalší model, v ktorom je zoznam značiek rozdelený podľa súborov, sme vytvorili funkciu, ktorá ,,zgrupí“ značky s rovnakým Id súboru do jedného zoznamu. Takto získame zoznam, ktorý obsahuje pre každý záznam samostatný zoznam značiek, ktoré sú ,,zgrupené“ pre konkrétny súbor. Tento zoznam spracujeme a naplníme náš finálny zoznam na základe modelu, ktorý obsahuje meno, Id súboru a zoznam značiek v danom súbore.

Výsledný zoznam ktorý sme získali popísaným procesom spracovania je následne jednoduché zobraziť v *parciálnom view*, kde je zobrazenie značiek roztriedené podľa súborov a ku každej značke je pridaný odkaz na súbor a riadok, kde sa nachádza.

Testovanie

Testovanie implementovaného filtra prebiehalo pomocou porovnania pri voľbe všetkých autorov a značiek voči konkrétnej voľbe pomocou filtra. Pozorovali sme, či sa správne zúžil rozsah a nič sa nestratilo, prípadne niečo nebolo navyše. Môžeme povedať, že filter prebehol na základe zvolených a následne spracovaných vstupných údajov správne.

15.14 Automatická aktualizácia počtu neprijatých notifikácií

Počet neprečítaných správ notifikácií sa aktualizuje až po novom načítaní celej stránky. Aktualizácia počtu sa však musí vykonať už po kliknutí na neprečítanú správu v menu.

Návrh

Pri zobrazení dialógu je potrebné vykonať kontrolu, či nebolo kliknuté na neprečítanú správu. Tento počet je potom potrebné znížiť.

Riešenie

Na zníženie počtu neprečítaných notifikácií sme použili funkciu pomocou JavaScriptu, ktorá po kliknutí na odkaz skontroluje, či nebolo kliknuté na odkaz s neprečítanou správou. Ak áno, pomocou skriptu sa zníži počet neprečítaných správ zobrazený pri obálke a vymaže sa text ,,New“ pred odkazom na správu.

Testovanie

Testovanie prebiehalo vizuálnou kontrolou prostredníctvom interakcie s odkazmi na zobrazenie dialógov k jednotlivým správam notifikácií. Počet neprečítaných notifikácií sa znižoval korektne, pričom sa aj upravil odkaz z ktorého bola odstránená fráza ,,New“ informujúca o novej značke.

15.15 Zobrazenie "loaderu" pri načítaní zoznamu značiek

Načítanie zoznamu značiek trvá dlhšiu dobu, preto je potrebné na stránke zobraziť ,,loader“, ktorý informuje používateľa o prebiehajúcom spracovaní údajov.

Analýza

S implementáciou ,,loaderu“ sa môžeme stretnúť na viacerých miestach v systéme. Jeden z nich je umiestnený aj pri zobrazení štatistík používateľa, pričom sa zobrazuje ako krútiace koliesko na obrázku formátu gif, kým sú spracovávané údaje na pozadí.

Návrh

Pri našom riešení navrhujeme použiť overenú, už implementovanú funkcionalitu v systéme, preto podobný ,,loader“ ako pri štatistikách aplikujeme na náš zoznam značiek s filtrom.

Riešenie

Prvou úlohou pri implementácii ,,loaderu“ bolo rozdelenie stránky na dve časti. Jedna časť obsahuje možnosti výberu, čiže filter, druhá časť obsahuje údaje, ktoré sa majú načítať a kvôli ktorým je potrebné zaviesť ,,loader“, kým sa tieto údaje získavajú. Zo stránky zobrazenia filtra a zoznamu značiek sme preto oddelili zoznam značiek do samostatného *parciálneho view*. Dôležitým rozdielom je aj presunutie volania funkcie pre získanie filtrovaných značiek z *controlleru* hlavného *view* do *controlleru parciálneho view,* vďaka čomu je možné aplikovať ,,loader“, keďže sa zobrazí na hlavnej stránke počas čakania na výsledky z *parciálneho view*.

Stránka tak pozostáva z filtra, ktorý už nie je iba klasickým HTML spracovaním stlačenia tlačidla a vybraných možností, ale implementovali sme AJAX funkcionalitu, ktorá po stlačení odovzdá vybrané údaje *dropdown listov* pre filtroranie *parciálnemu view*, na pozadí sa vykonajú určité skripty, ktoré počas načítania údajov v *controlleri parciálneho view* zobrazia ,,loader“ a po načítaní údajov sa zobrazí *parciálne view* so zoznamom značiek na rovnakej stránke pod možnosťami výberu filtrácie. Vďaka tomuto rozdeleniu nie je potrebné načítať nanovo celú stránku s filtrom a zoznamom značiek, ale iba sa aktualizuje *parciálne view* obsahujúce zoznam značiek po aplikácií filtra. Týmto spôsobom sa dokonca vyriešil aj problém zachovania vybranej možnosti filtra iným spôsobom od toho, ktorý sme implementovali v predchádzajúcom šprinte.

Výsledný filter s prebiehajúcim ,,loaderom“ môžeme vidieť na obrázku č. 102.

Obr. č. 102: ,,Loader“ na pravej strane pri načítaní filtrovaného zoznamu značiek

Testovanie

Testovanie sme vykonali vizuálne. Pokiaľ prebiehalo načítanie zoznamu značiek, zobrazil sa krútiaci ,,loader“, čiže sme boli informovaný o prebiehajúcom spracovávaní údajov.

Okrem tejto funkcionality sme skontrolovali aj zoznam získaných značiek, keďže sme stránku so zoznamom značiek rozdelili na dve časti. Zoznam značiek bol identický ako v prípade, keď stránka

bola v jednom celku, čiže pri implementácií *parciálneho view* nedošlo k chybám a strate údajov voči pôvodnému riešeniu.

15.16 Filter CR a AST projektov v správe repozitárov a filter projektov a používateľov v správe používateľov projektu

Zákazník požaduje aby dané tabuľky obsahovali filter pre jednoduchšie prehľadávanie.

Riešenie

Čísla changesetov: 1464,1466, 1474

JavaScript: CodeReview.Web.Scripts.available_filter.js

View:

CodeReview.Web.Views.ProjectManager.Swapper

CodeReview.Web.Views.ProjectManager.SwapperUser

Testovanie

Vstup: Písanie do filtra. sledovanie zmien zoznamu

Výstup: Sledovanie zmien v rozhraní a kontrola s databázou.

15.17 Zobraziť mená používateľov v správe práv používateľov projektu

Zákazník požaduje aby boli pri správe práv používateľov zobrazené mená používateľov a nie ich AIS loginy pre lepšiu prehľadnosť

Riešenie

Čísla changesetov: 1340

View:

CodeReview.Web.Views.Users. EditorTemplates.

Testovanie

Vizuálna kontrola a kontrola porovnaním s databázou

15.18 Usporiadanie zoznamov podľa abecedy

Zákazník požaduje aby zoznamy v správe repozitárov, v správe používateľov projektu, v správe administrátorov, v správe práv používateľov, projektoch a zozname používateľov na úvodnej stránke, boli usporiadané, pre zlepšenie prehľadnosti.

Návrh

Zobrazované listy boli zoradené pomocou OrderBy.

Riešenie

Čísla changesetov: 1340,1341,1342,1464

Controller:

CodeReview.Web.Controllers.UsersController

CodeReview.Web.Database.UserMethods

CodeReview.Web.Database.ProjectMethods

Model:

CodeReview.Web.Models.UsersRolesModel

View:

CodeReview.Web.Views.ProjectManager.Swapper

Testovanie

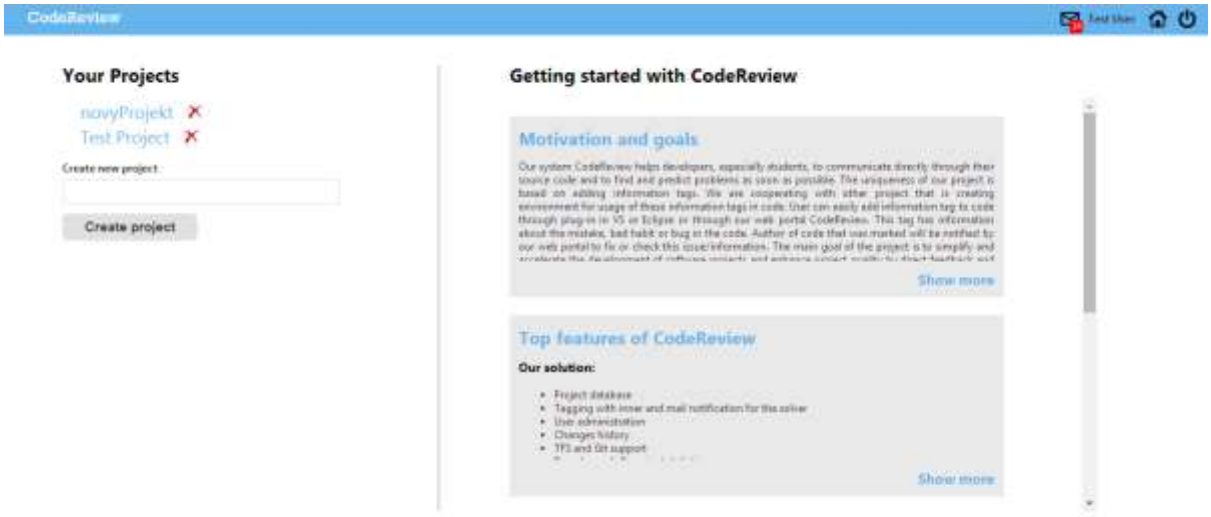
Vizuálny test vybraných zoznamov, kontrola zobrazených dát s databázou.

15.19 ZMENA DIZAJNU DOMOVSKEJ STRÁNKY PROJEKTOV

(Product Backlog Item 873): Úlohou bola implementovať zmeny domovskej stránky, ktoré boli navrhnuté v predchádzajúcom šprinte.

Riešenie

Na obrázku č. 103 je znázornená nová domovská stránka projektov. Táto stránka so zobrazuje používateľovi po prihlásení do systému. Na ľavej strane sa nachádzajú projekty používateľa. Na pravej strane sa zobrazia základné informácie o projekte a informácie o našom tíme.



Obr. č. 103 Nová úvodná stránka projektov

15.20 Nahradenie stránky Under construction domovskou stránkou projektov

(Product Backlog Item 870): Používateľovi sa po prihlásení zobrazilo úvodná stránka s obrázkom *„Under Constraction“*.

Riešenie

Táto stránka sa nahradila novou úvodnou stránkou pre projekty, ktorá je znázornená v na obrázku č. 103 v predchádzajúcej kapitole.

15.21 Context menu v strome je prekryté kódom

Riešenie

Upravili sme štýly pre context menu tak aby sa zobrazovalo „vyššie“. (doplnený z-index)

15.22 Ošetrenie pridania prázdneho aliasu (prázdny string)

(Bug 1074): Pri pridávaní aliasov bolo možné pridať aj prázdny reťazec.

Riešenie

Tento problém bol odstránený overením dĺžky reťazca. Systém umožňuje pridanie aliasu, ktorý sa skladá aspoň z 2 znakov. Do tejto dĺžky nie sú pripočítané prázdne znaky ako napr. medzery. Na obrázku č. 104 sa nachádza spätná správa pre nesprávny vstup.

Obr. č. 104 Nesprávny vstup pre alias

15.23 [BUG] Značky sa nezobrazujú korektne

Umiestnenie značiek pri zobrazení v súboroch je nesprávne

Riešenie

Prvým krokom riešenia bolo sprehľadnenie zdrojového kódu. To prebiehalo tým spôsobom, že všetok kód z CSHTML súborov, ktorý obsahoval definíciu štýlu, sme vybrali a uložili do CSS súboru.

Následne sme preskúmali a vyriešili problémy pri zobrazení značiek, ktorými bolo napríklad nesprávne vypočítanie umiestnenia značky od vrchu okna, resp. elementu v ktorom je zobrazená, pričom bolo potrebné preskúmať a poupravovať vlastnosti niektorých elementov pri zobrazení.

Testovanie

Testovanie zobrazenia prebiehalo vizuálnou kontrolou, pričom sme skúmali správne umiestnenie elementov značiek.

16 ŠTVORLÍSTOK

Číslo šprintu: 10

Začiatok šprintu: 28.04.2014

Koniec šprintu: 19.05.2014

Príbehy:

- **Notifikácia sa pri domovskej stránke projektu zobrazila ako hlavná stránka (nie dialóg)**
- **Po prechode na adresu vygenerovanú notifikáciou je odmietnutý prístup**
- **Pri zobrazení zmien vo zvolených odovzdaniach súboru sa zobrazí aj jeden vybratý súbor**
- **Zobrazenie autora pomocou Display Name pri značkách**
- **Upraviť zobrazenie All Notification Messages - zaradenie, show more**
- **Zle poradie notifikácii vo vyskakovacom "menu" a aj keď sa ďalej klikne na "See all". "See all" treba prepisat na "Show all“**
- **Zobrazenie display name pri správe používateľov projektu**

16.1 Notifikácia sa pri domovskej stránke projektu zobrazila ako hlavná stránka (nie dialóg)

Pri zobrazení notifikácie na domovskej stránke projektu (Home) sa notifikácia nezobrazí v na to určenom dialógu ale zobrazí sa priamo v obsahu stránky.

Analýza

Problém sa prejavoval iba pri zobrazení v systéme na servery. Na lokálnom úložisku sa problém neprejavoval.

Riešenie

Chybu sa nám nepodarilo odstrániť z dôvodu, že chyba sa prejavovala iba na serverovej strane takže nebolo možné zistiť z akého dôvodu sa udalosť deje.

Obr. č. 106 Chybné zobrazenie notifikácie

16.2 Po prechode na adresu vygenerovanú notifikáciou je odmietnutý prístup

Zákazník požaduje, aby bola adresa v notifikácii plne funkčná a aby bol prístup do projektu povolený.

Analýza

Problém vznikol pri generovaní adresy pri prijatí notifikácie. Je potrebné presne určiť, ku ktorému projektu sa samotná značka vzťahuje.

Návrh

V tomto prípade bude potrebné skontrolovať databázový model a následne stačí zistiť, aký projekt v našom systéme sa viaže na značku. Následne zistíme jeho ID a toto ID potom vložíme do vygenerovaného odkazu.

Riešenie

Samotné riešenie bolo vytvorené na základe návrhu. Samotná značka sa viaže na notifikačný filter a každý filter sa viaže presne na jeden určitý projekt v našom systéme. Riešenie teda spočívalo vo vytvorení správnych databázových dopytov. Na základe notifikačného filtra a jeho vlastnosti „EventListenerId“ sme zistili aj k akému projektu sa značka vzťahovala. Následne sme už len zistili presné ID tohto projektu a toto ID sme vložili do vygenerovaného odkazu.

Pôvodný problém bol spôsobený vložením nesprávneho ID projektu. Používateľ nemal prístup k takémuto projektu a systém mu odmietol prístup. Tento problém bol odstránený.

Testovanie

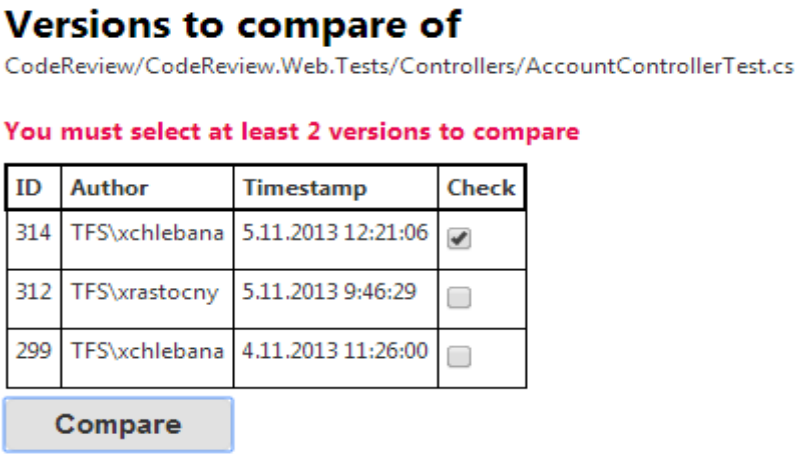
Testovanie prebiehalo manuálne, pretože systém notifikácií je plne funkčný iba na našom vzdialenom produkčnom servery. Server značiek (ITM) dokáže kontaktovať iba náš produkčný server a preto sa notifikácie zobrazujú korektne iba tam. Samotné notifikácie mali pri testovaní funkčný, klikateľný odkaz, ktorý správne odkazoval na projekt a repozitár v ktorom bola značka pridaná.

16.3 Pri zobrazení zmien vo zvolených odovzdaniach súboru sa zobrazí aj jeden vybratý súbor

(Bug 1122): Pri zobrazení zmien vo zvolených odovzdaniach bolo možné zvoliť aj jedno odovzdanie.

Riešenie

Bola pridaná podmienka pomocou *JavaScript*, ktorý sleduje počet zvolených odovzdaní na zobrazenie. Pri nesprávnom počte sa zobrazí chybová správa. Jedno odovzdanie je možné vybrať iba v prípade, že súbor má iba jednu verziu. Na obrázku č. 105 je znázornená zvolenie nesprávneho počtu verzií pre porovnávanie.



Obr. č. 101 Nesprávny počet zvolených verzií na porovnávanie

16.4 Zobrazenie autora pomocou Display Name pri značkách

Pri značkách je potrebné zobrazovať *DisplayName* a nie *MasterAlias*, *login* a podobne.

Riešenie

Predpokladáme, že po predchádzajúcom spracovaní je v značke ako autor priradený *MasterAlias*. Pri úprave autora značky sme v tomto kroku pridali funkciu, ktorá slúži na nahradenie *MasterAliasu* za *DisplayName*. Ak pred úpravou v značke nie je uložený predpokladaný *MasterAlias* z ešte predchádzajúceho spracovania autorov alebo v databáze nemáme potrebný údaj, potom nie je možné správne priradiť *DisplayName* a v značke necháme pôvodný údaj o autorovi.

Testovanie

Testovanie prebehlo vizuálnou kontrolou, pričom sme pozorovali či sa *MasterAlias* vhodne nahradil údajom *DisplayName*.

16.5 Upraviť zobrazenie All Notifcation Messages - zaradenie, show more

Pri zobrazení všetkých správ notifikácií nie sú tieto správy nijak usporiadané. Tlačidlo ,,Show more“ na konci stránky sa taktiež zobrazuje donekonečna.

Návrh

Pre správne poradie bude potrebné upraviť ich poradie pri zobrazení, prípadne už pri získavaní.

Pre tlačidlo ,,Show more“ zavedieme limit, dokedy sa má zobrazovať.

Riešenie

Pri získavaní správ notifikácií z databázy sme zadali do volania možnosť, ktorá získané položky usporiada. Usporiadanie je vykonané na základe dátumu vytvorenia správy notifikácie. Takto upravený zoznam sa následne korektne zobrazí na stránke.

Pri tlačidle ,,Show more“ sme zaviedli kontrolu, či rozsah zobrazenia správ nie je rovnaký alebo neprekročil počet všetkých správ. Ak nastane takáto situácia, tlačidlo sa už nezobrazí.

Testovanie

Testovanie prebehlo najmä po vizuálnej stránke. Pri tlačidle ,,Show more“ sme tiež overili, či sa zobrazili všetky správy a tlačidlo sa prestalo zobrazovať, keď už neboli ďalšie správy pre zobrazenie.

16.6 Zle poradie notifikácii vo vyskakovacom "menu" a aj keď sa ďalej klikne na "See all". "See all" treba prepísať na "Show all“

Riešenie

Pri načítaní notifikácií z databázy sme ich poradie upravili tak aby sa ako prvé zobrazovali najnovšie štatistiky.

Následne sme zmenili položku v menu notifikácii z „See all“ na „Show all“.

16.7 Zobrazenie d*isplay name* pri správe používateľov

Zákazník požaduje aby boli pri správe práv používateľov zobrazené mená používateľov a nie ich AIS loginy pre lepšiu prehľadnosť.

Riešenie

Čísla changesetov: 1675

Controller:

CodeReview.Web.Database.UserMethods

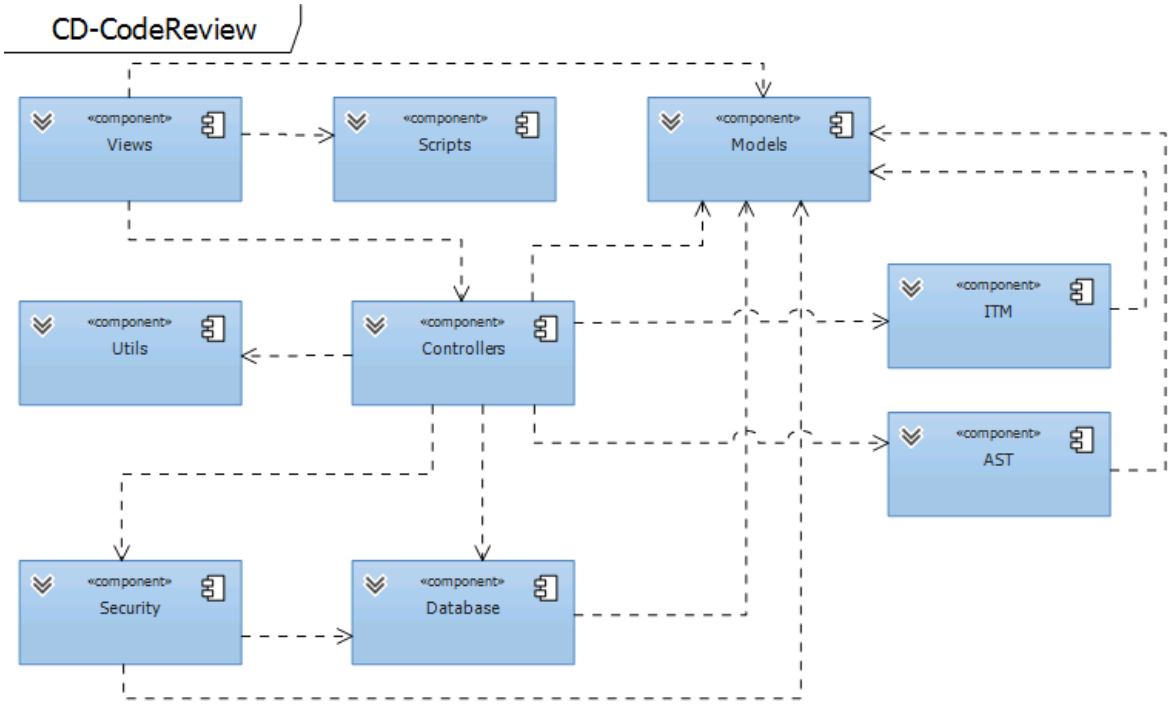
Testovanie

Vizuálna kontrola a kontrola porovnaním s databázou.

17 CELKOVÝ POHĽAD

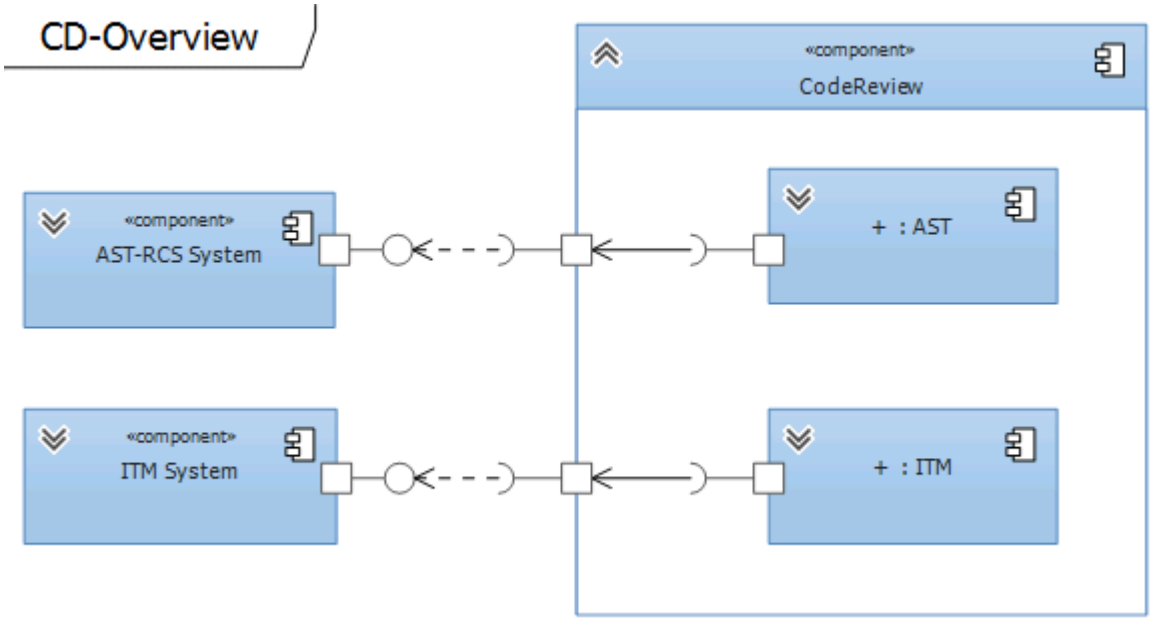
V tejto kapitole je zobrazený pohľad na softvér ukončení dvoch semestroch vývoja. Sú tu uvedené „component diagramy“ a „use-case“ diagram systému ako celku. V „component diagramoch“ sú použite iba „<<use>> dependency“ vzťahy. Táto kapitola obsahuje aj konečný dátový model databázy a výslednú mapu vytvoreného webového portálu.

„Component diagram“ celého systému je zobrazený na obrázku č. 107.



Obr. č. 107 "Component diagram" CodeReview

„Component diagram“ zobrazujúci komunikáciu medzi komponentmi CodeReview a inými systémami (ASTRCS a ITM) je zobrazený na obrázku č. 108. Komponenty CodeReview AST a ITM spracúvajú dáta zo systémov ASTRCS a ITM. Dáta sú ďalej posielané do „controllerov“.



Obr. č. 108 Interakcia komponentov CodeReview s inými systémami

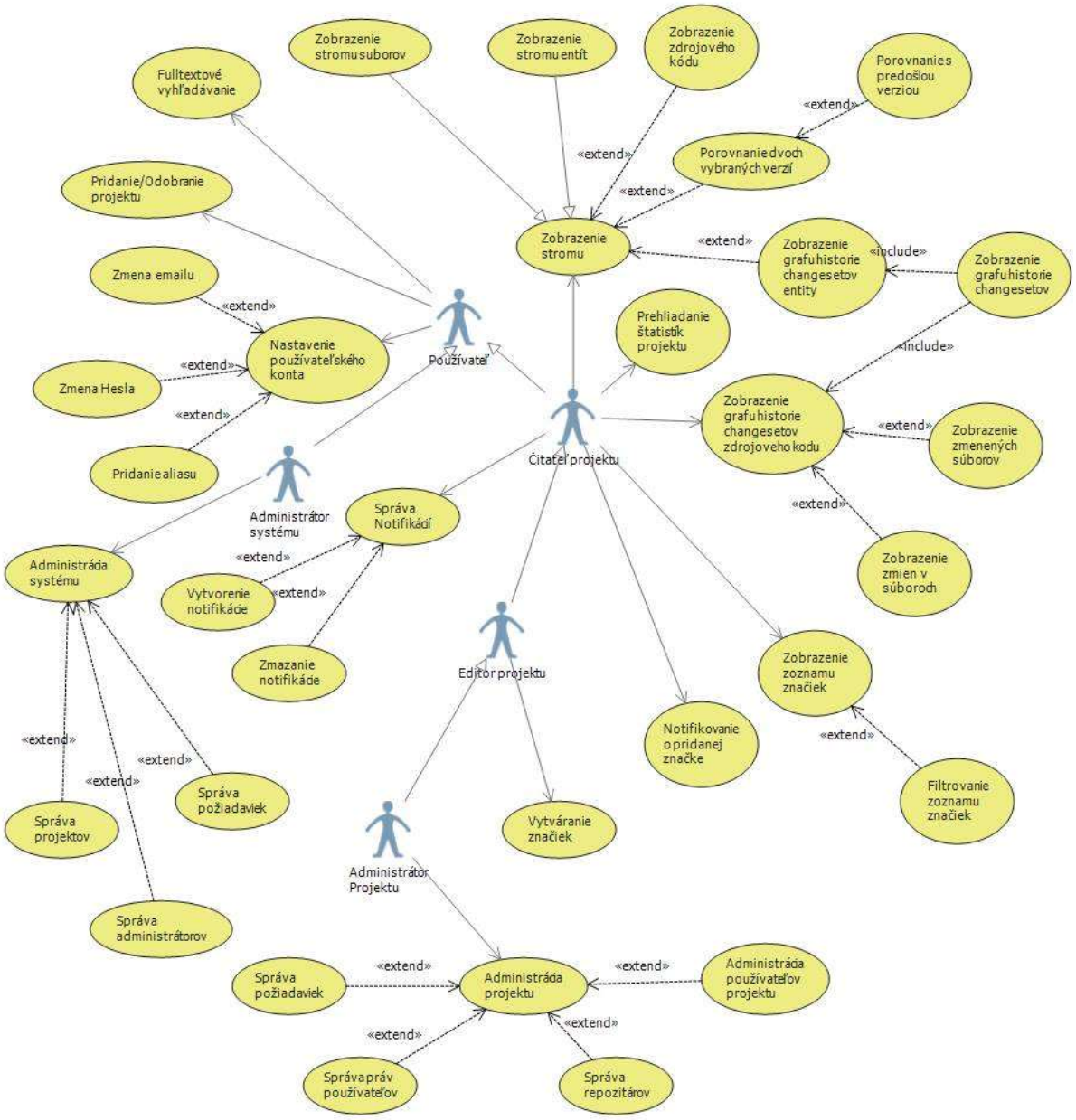
„Use Case diagram“ celého systému je zobrazený na obrázku č. 109. Na diagrame môžeme vidieť jednotlivé používateľské role a aké prípady použitia k ním patria. Systém ma dve základné roly a to používateľ systému a administrátor systému. Používateľ systému však je bližšie špecifikovaný rolami ktoré ma na projekte a to administrátor projektu, editor projektu a čitateľ projektu. Niektoré prípady sú vykonateľné na základe stereotypov *extend* a *include*.

Medzi hlavné prípady použitia patria:

- Nastavenie používateľského konta - Používateľovi je umožnené aby si modifikoval nastavenia svojho konta podľa jeho potreby, a to v tejto fáze systému najmä pridával aliasy, menil heslá alebo zmenil emailovú adresu.
- Administrácia systému - je určená pre rolu administrátora systému ktorému je umožnené aby spravoval projekty a tiež repozitáre, ktoré k daným projektom patria, pokiaľ používateľ požiada o pridanie repozitára administrátor mu daný repozitár pomocou správy projektov pridá. Administrátor ma tiež právo meniť administrátorov systému pomocou správy administrátorov. Administrácia systému tiež umožňuje zobraziť všetky požiadavky, ktoré boli používateľmi do systému pridané, schváliť ich alebo odmietnuť.
- Správa notifikácií - umožňuje, aby používateľ mal možnosť pridávať si rôzne typy značiek k svojmu projektu, popísať ich, editovať, nastaviť si ich podľa svojej potreby.
- Administrácia projektu - je určená pre rolu administrátora projektu, ktorý môže v systéme meniť používateľov priradených k projektom, ktorých je administrátor, taktiež má možnosť meniť práva týchto používateľov v zadanom projekte. Administrácia projektu umožňuje taktiež požiadať o pridelenie repozitára a prehliadanie požiadaviek, ktoré administrátor projektu zadal a sledovať v akom sú stave.
- Zobrazenie stromu súborov a entít - umožňuje zobrazenie zdrojového kódu, porovnanie rôznych verzií projektu, či zobrazenie grafu histórie changesetov s farebne zvýraznenými zmenami, podľa ich autorov.
- Prehliadanie štatistík projektu - umožňuje získavanie informácií o používateľoch a ich činnosti v systéme za dané časové obdobie.

- Vytváranie značiek - dôležitá časť systému je práve vytváranie značiek, používateľ má možnosť kdekoľvek v kóde kliknutím umiestniť značku, vložiť jej popis a typ, pričom po jej uložení budú ostatní používatelia priradený k danému projektu notifikovaní o jej pridaní.
- Zobrazenie zoznamu značiek - používateľ je notifikovaný o pridaní značky a po prihlásení ich vidí vo vyskakovacom menu po kliknutí na obálku. Značky sú zobrazené v časovom slede od najnovšej po najstaršiu. Používateľ má možnosť si všetky značky zobraziť spolu s detailnejším popisom.

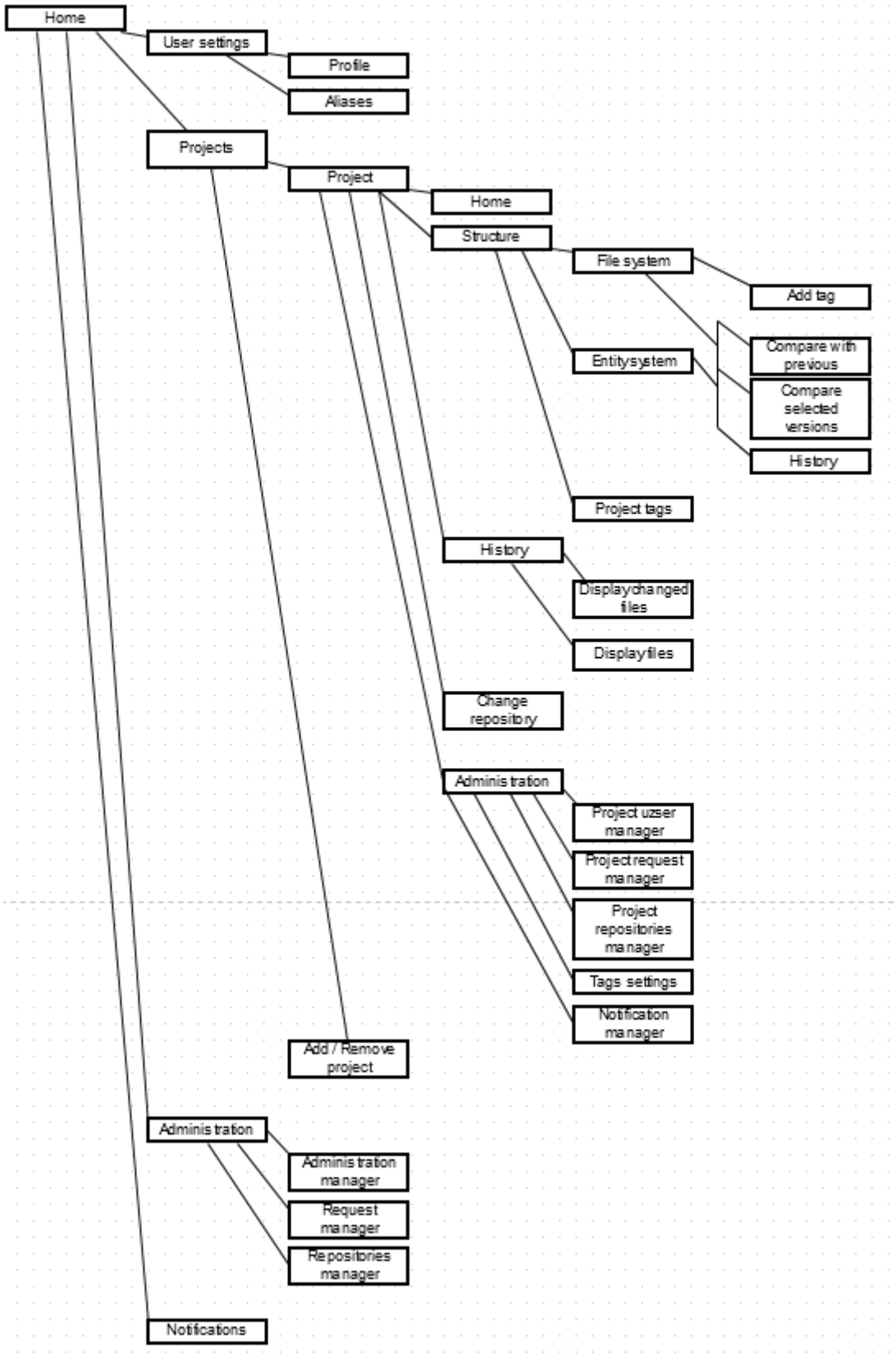
Celkový diagram by sa mohol v budúcnosti ďalej rozširovať pridávaním dodatočnej funkcionality, napr. možnosť pridávať komentáre ku značkám alebo zdrojovým súborom.



Obr. č. 109 Kompletný Use Case diagram.

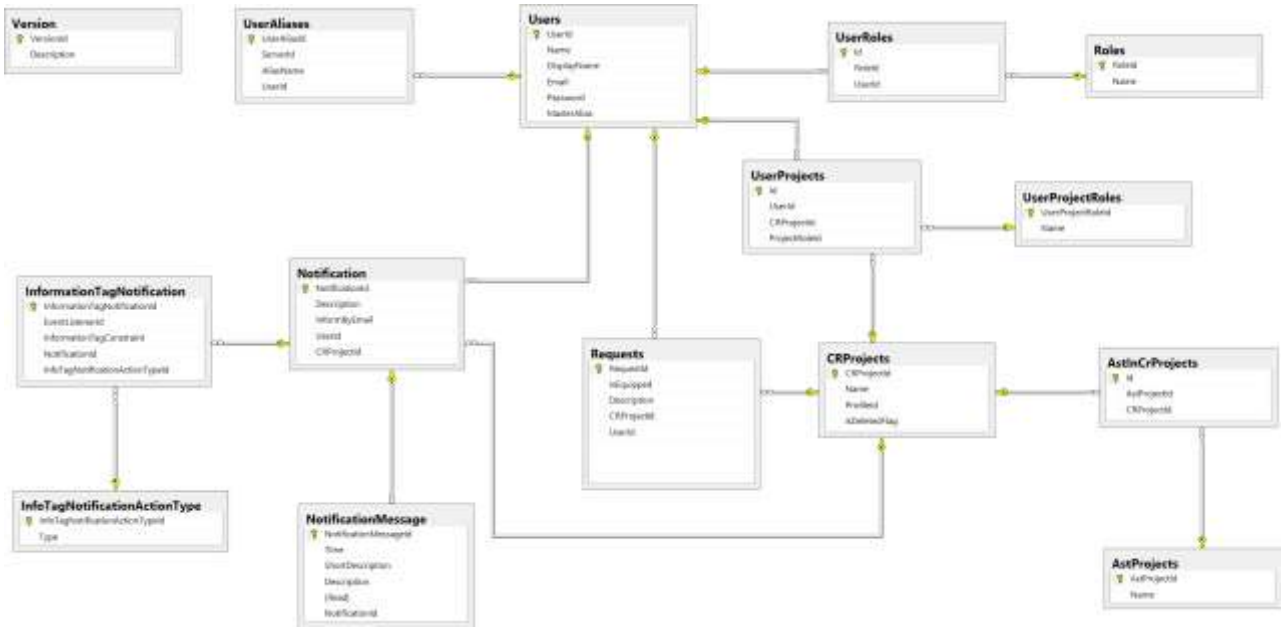
Mapa stránky projektu je zobrazená na obrázku č. 110. Táto mapa vyjadruje, akým spôsobom je možné pohybovať sa v rámci nášho webového rozhrania. Samotná rozhranie je teda rozdelené na 4 najhlavnejšie časti:

- User settings – nastavenia používateľského konta
- Projects – najhlavnejšia funkcionalita; prístup k projektom a k jednotlivým repozitárom
- Administration – administračný panel prístupný len pre systémového administrátora
- Notifications – príjem systémových notifikácií o pridanej značke



Obr. č. 110 Konečná mapa stránky CodeReview.

Na obrázku č. 111 je zobrazený fyzický databázový model systému CodeReview. Databázový model celkovo obsahuje 20 tabuliek. Tabuľka *Version* sa používa na identifikáciu aktuálnej verzie databázy a podľa jej záznamov sa identifikuje, či sa má aplikovať migrácia. Tabuľky *UserAliases, Users, UserRoles a Roles* sa používajú pre identifikovanie používateľov v našom systéme. Ostatné tabuľky definujú funkcionalitu systému.



Obr. č. 111 Konečný fyzický databázový model systému CodeReview

Používateľská aj systémová príručka sú súčasťou tohto dokumentu vo forme príloh a sú presne popísané v kapitolách 20 a 21.

18 ČO SME NESTIHLI

Na projekte CodeReview sme pracovali jeden akademický rok, pri čom sme vytvorili systém na správu prehliadok kódu. Používatelia môžu vytvárať projekty, prehliadať zdrojové kódy, pridávať značky a byť o nich notifikovaný.

Vytvorený systém obsahuje ešte menšie nedostatky, ktoré by sa ľahko odstránili dodatočným testovaním. Testovanie bolo vykonané aj po ukončení vývoja, pravdaže neboli odhalené všetky chyby. Určite by bolo treba minimálne mesiac testovania na odhalenie väčšiny chýb, ktoré ešte existujú v systéme. Okrem nedostatku testovania môžeme povedať že sme stihli implementovať množstvo funkcií. Určite by sa však našli nové funkcie, ktoré by pomohli systému byť priatelenejším pre používateľa a ponúknuť mu kvalitnejšie prostredie.

Pri podobnom projekte sú nové funkcie vymýšľané častokrát počas implementácie. Tak to bolo aj v tomto prípade počas letného semestra sme vymysleli a opísali mnohé funkcie, ktoré by mohli byť implementované v budúcnosti. V prípade že by sme pokračovali s projektom, určite by pribudlo veľa nových zaujímavých príbehov na implementáciu. Medzi možné rozšírenia do tohto projektu patrí:

- Pridávanie komentárov ku zdrojovým súborom
- Pridávanie komentárov ku značkám
- Editovanie značiek
- Podpora ďalších verziovacích systémov (nie len TFS a Git)
- Vytváranie používateľských skupín
- A iné.

V budúcnosti je projekt možné rozšíriť viacerými spôsobmi a je len na požiadavkách product ownera, aké funkcionality budú pridané v budúcich iteráciách vývoja.

19 ČO SME SA NAUČILI

Počas práce na tímovom projekte sme sa naučili ako funguje tím v skutočnosti. Každý člen získal praktické vedomosti pri vykonávaniu manažérskej činnosti na pozícii, ktorú si vybral.

Oboznámili sme sa s hlavnými aspektmi agilného vývoja pomocou metódy *SCRUM*. Naučili sme sa ako dôležitá je komunikácia v tíme, postupy práce v tíme, ako plánovať jednotlivé úlohy. Stali sme sa lepším v plánovaní úloh a postupne sa nám podarilo lepšie odhadnúť stupeň zložitosti jednotlivých úloh a ich trvanie. Vďaka rôznym úlohám, ktoré sme vykonávali počas dvoch semestrov, sme boli schopní zistiť naše silné i slabé stránky vzhľadom na tímovú prácu aj na naše odborné znalosti.

Okrem vyššie spomenutých všetci členovia tímu získali znalosti v programovacích jazykoch *C#, JavaScript, HTML, CSS,* v technológií *.NET MVC4* ako aj skúsenosti so systémom *TFS*.

20 PRÍLOHA – POUŽÍVATEĽSKÁ PRÍRUČKA

20.1 PREDSTAVENIE PROJEKTU

Systém CodeReview slúži primárne na vykonávanie prehliadky zdrojových kódov v projektoch. Verzia ku dňu 12.5.2014 podporuje projekty v programovacích jazykoch C# a Java. V ďalších kapitolách dokumentu popíšeme jednotlivé časti systému a spôsob ich použitia.

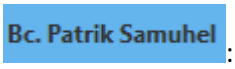
20.2 VSTUP DO SYSTÉMU

Systém CodeReview je vytvorený ako webová aplikácia dostupná priamo z webového prehliadača. Systém je použiteľný vo všetkých často používaných webových prehliadačoch. Systém je dostupný na adrese http://147.175.149.11/Codereview/. Po zobrazení úvodnej stránky je potrebné sa prihlásiť. Do systému je možné sa prihlásiť pomocou AIS loginu.

20.2.1 Úvodná obrazovka

Úvodná obrazovka obsahuje základné info o systéme a jeho častiach. Na navigáciu na stránke slúžia tlačidlá v pravom hornom rohu:

Popis jednotlivých položiek:

-  : notifikácie o pridaných značkách v systéme (bližšie popísané v kapitole 3.2.1)
-  : meno prihláseného používateľa
-  : zoznam projektov používateľa
-  : administrácia (iba pre užívateľov s administrátorskými právami)
-  : odhlásenie zo systému

20.2.2 Manažment projektov 

Každý používateľ má možnosť vytvoriť projekt v našom systéme (ďalej CRProjekt). Tento projekt môže obsahovať viacero programových projektov, ktoré sú v našom systéme nazvané „repozitár“.

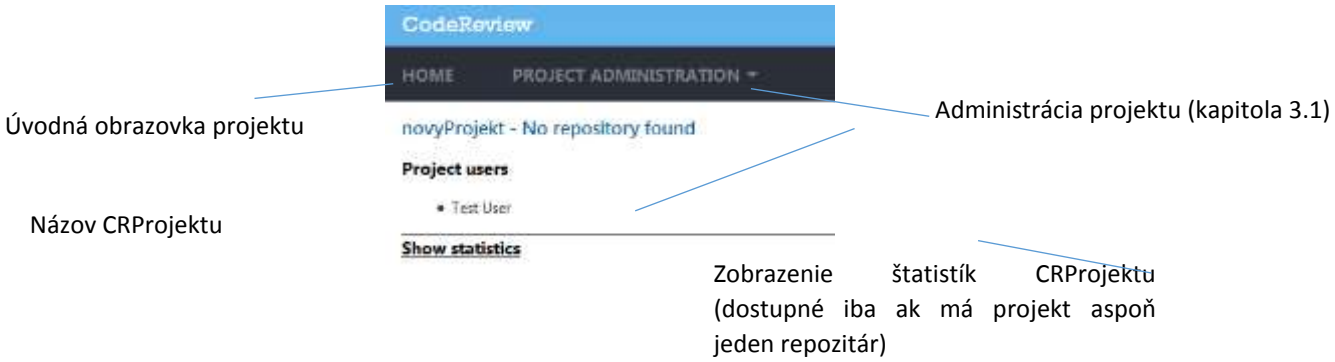
Postup vytvorenia nového projektu:

1. Do kolónky „Create new project“ zadaj názov projektu.
2. Stlačením tlačidla „Create project“ so nový projekt pridá do zoznamu „Your Projects“

V kapitole 3. popíšem možnosti práce s CRProjektom.

20.3 PRÁCA S PROJEKTOM

Po výbere projektu sa zobrazí úvodná obrazovka projektu:



Po pridaní CRProjektu je potrebné vložiť do projektu repozitáre. Repozitáre do projektu vkladá administrátor po obdŕžaní požiadavky. Bližšie informácie v kapitol 3.1.

20.3.1 Administrácia projektu

Menu „Project administration“ obsahuje všetky položky súvisiace s administráciu zvoleného CRProjektu:

Jednotlivé položky menu popíšeme v nasledujúcich kapitolách.

20.3.1.1 Project users

Používateľ, ktorý vytvoril projekt k nemu môže prideliť ďalších používateľov. Pridávanie prebieha nasledovne:

1. V kolónke „Projects“ zvolíme CRProjekt ku ktorému chceme pridať používateľov.
2. V kolónke „Available“ zvolíme všetkých používateľov, ktorých chceme ku projektu pridať podržaním klávesy „Ctrl“ a kliknutím na mená používateľov.
3. Stlačením tlačidla „Add“ pridáme k projektu zvolených používateľov.
4. Zmeny sa uložia stlačením tlačidla „Save“

Odstránenie používateľov z projektu prebieha nasledovne:

1. V kolónke Assigned zvolíme používateľov, ktorých chceme z projektu odstrániť
2. Stlačením tlačidla „Remove“ odstránime používateľov z projektu

3. Zmeny sa uložia stlačením tlačidla „Save“

20.3.1.2 Request a repository

Ako sme už skôr spomenuli, k projektu je potrebné požiadať o pridelenie repozitárov. Systém podporuje repozitáre uložené v systémoch TFS a Git.

Prioritne sú repozitáre uložené v systéme AST-RCS. V prípade že systém AST-RCS repozitár neobsahuje je potrebné poslať požiadavku musí používateľ vyplniť požiadavku na konkrétny TFS/Git repozitár.

Po zvolení/vyplnení údajov o repozitári sa tlačidlom „Send“ odošle požiadavka pre administrátora.

20.3.1.3 Show requests

Používateľ má možnosť zobrazenia požiadaviek o repozitáre a ich aktuálny stav:

20.3.1.4 Manage project users

Používateľ, ktorý vytvoril projekt má možnosť udávať privilégia ostatným používateľom. Popis privilégií je nasledovný:

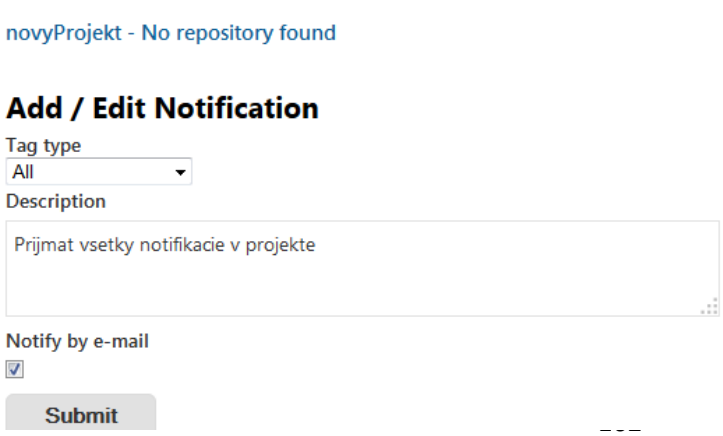
- Reader: iba vidí repozitáre a ich zdrojové kódy,
- Editor: ako Reader + môže pridávať značky ku zdrojovým kódom
- Administrator: ako Reader a Editor + môže meniť práva používateľov projektu

20.3.1.5 Tag settings

Nastavenie profilu značiek. Profil značiek obsahuje rôzne značky, ktoré je možné do zdrojových kódov pridávať. Používateľ má na výber z niekoľkých profilov. Po výbere profilu značiek sa zmeny uložia stlačením tlačidla BaseProfile.

20.3.1.6 Manage notifications

Používateľ si môže nastaviť o akých značkách chce byť notifikovaný. Po stlačení tlačidla „Add new notification“ sa zobrazí okno pre vytvorenie podmienky:



„Tag type“ obsahuje informáciu o type značky, o ktorej chceme prijímať notifikácie.

„Description“ je opis podmienky.

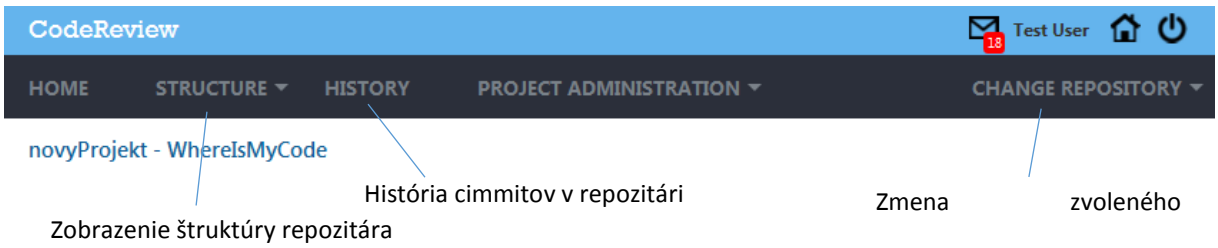
O notifikáciách je možné byť informovaný aj prostredníctvom e-mailu.

Tlačidlom „Submit“ používateľ vytvorí podmienku.

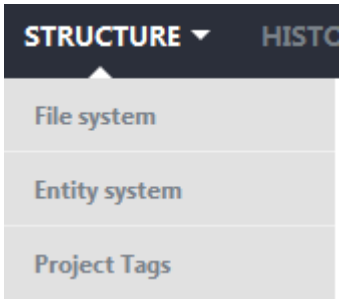
Používateľ môže pridať ľubovoľné množstvo podmienok.

20.3.2 Práca s projektom

Po pridaní repozitárov do projektu sa zobrazia nové položky do menu na úvodnej stránke projektu:



„Structure“ obsahuje položky zobrazenia obsahu projektu rôznymi metódami:



„File system“ – zobrazenie stromu súborov

„Entity system“ – zobrazenie entít (trie, metód, ...) repozitára

„Project Tags“ – zobrazenie značiek vložených do nášho systému

20.3.2.1 File system

Strom súborov obsahuje všetky súbory v najnovšej verzií projektu(repozitára).

20.3.2.1.1 Scenáre použitia

Zobrazenie zdrojového kódu

Pre zvolený súbor sa zobrazí zdrojový kód na pravej strane stránky. Postup je nasledovný:

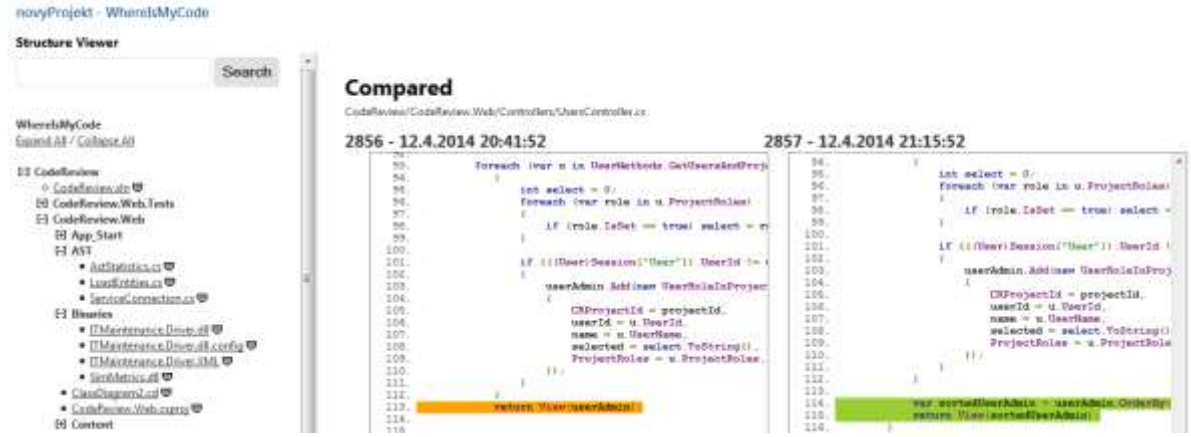
1. Rozbaľte strom súborov až po žiadaný súbor
2. Kliknite na názov súboru pre zobrazenie
3. Ak je to možné, súbor sa zobrazí na pravej strane stránky

Zobrazenie naposledy vykonaných zmien v súbore

Pre zvolený súbor je možné zobraziť porovnanie s poslednou verziou so zvýraznenými časťami zmeneného kódu.

Postup zobrazenia poslednej zmeny v súbore:

1. Rozbaľte strom súborov až po žiadaný súbor
2. Kliknite na tlačidlo kontextového menu vedľa názvu súboru
3. Zvoľte položku „Compare with previous“
4. Na pravej strane sa zobrazí aktuálna a druhá najnovšia verzia súboru



Zobrazenie rozdielov v špecifických verziách rovnakého súboru

Postup zobrazenia rozdielov v špecifických verziách súboru:

1. Rozbaľte strom súborov až po žiadaný súbor
2. Kliknite na tlačidlo kontextového menu vedľa názvu súboru
3. Zvoľte položku „Compare selected versions“
4. Zo zobrazenej tabuľky zvoľte dve verzie súboru
5. Stlačením tlačidla „Compare“ sa zobrazia zmeny vo zvolených verziách súboru

Zobrazenie histórie commitov zvoleného súboru

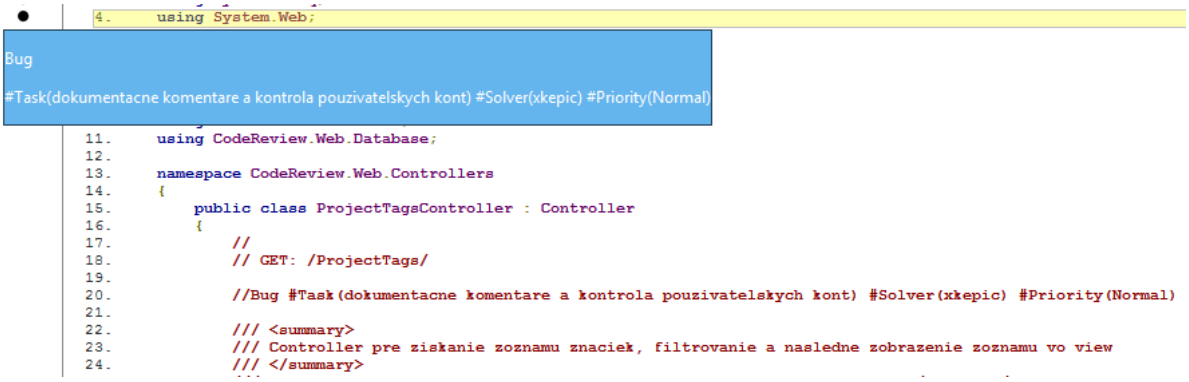
Postup zobrazenia histórie commitov v súbore:

1. Rozbaľte strom súborov až po žiadaný súbor
2. Kliknite na tlačidlo kontextového menu vedľa názvu súboru
3. Zvoľte položku „History“
4. Zobrazí sa obrazovka obsahujúca Branch graph commitov zvoleného súboru a popisi commitov.

Pridanie značky do zdrojového kódu

Hlavnou súčasťou nášho systému je možnosť pridania značiek do zdrojového kódu. Značky slúžia na označenie chyby alebo činnosti, ktorú treba v kóde vykonať. Každý profil značiek obsahuje niekoľko zvolených značiek, s presne určeným obsahom. V nasledujúcom postupe popíšeme postup pridania značky ku odhalenej chybe v kóde:

1. Zobrazíme si zdrojový kód súbory, v ktorom sa chyba nachádza
2. Označenie:
    a. riadok s chybou: klikneme ľavým tlačidlom myši na riadok v ktorom sa nachádza chyba
    b. chybná časť kódu: ľavým tlačidlom chyby označíme celu časť chybného zdrojového kódu
3. Po označení sa zobrazí tlačidlo „Add tag“
4. Stlačením tlačidla „Add tag“ sa zobrazí dialógové okno pre pridanie značky.
5. Z prvého Combo-boxu vyberieme typ značky „Bug“
6. Vyplníme ostatné položky, ktoré sú obsahom značky (opis chyby; kto má chybu vyriešiť; priorita odstránenia chyby; doplnkový komentár)
7. Stlačením tlačidla „Submit“ vložime značku do kódu (značka je viditeľná iba v našom systéme)
8. Po znovunačítaní stránky sa v ľavej časti kódu zobrazí značka, ktorú sme práve pridali

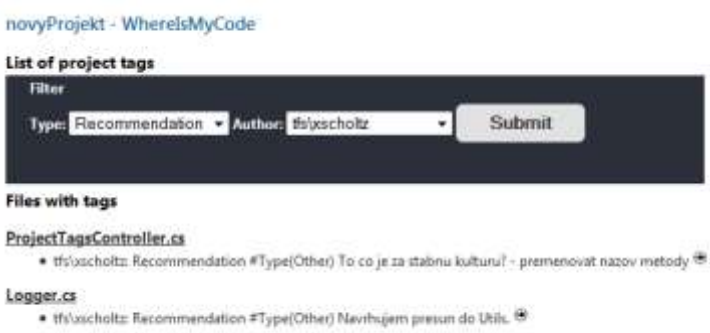


20.3.2.2 Entity system

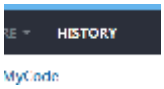
Entity system zobrazuje strom všetkých entít (classy, metódy, sripty ...) v repozitári. Strom zobrazených entít má rovnaké funckie ako strom „File system“

20.3.2.3 Project tags

Položka „Project tags“ zobrazuje značky pridané do aktuálneho repozitára. Fiter umožnuje zobraziť iba niektoré typy značiek od určitého autora. Na kód, v ktorom je značka pridaná je možné sa priamo presmerovať použitím šípky na pravej strane popisu značky.



20.3.2.4 History



Položka „History“ zobrazí históriu commitov v projekte (repozitári). História zmien sa zobrazí formou Branch graphu spolu s popisom každého commitu. Každý používateľ, ktorý vykonal commit je označený inou farbou. Stlačením tlačidla „Show“ v pravom dolnom rohu sa zobrazia commity všetkých vyššie označených používateľov.

20.3.2.4.1 Scenáre použitia

Zobrazenie zmien v zvolenom odovzdaní

V každom commite je možne zobraziť súbory, v ktorých bola vykonaná zmena. Postup zobrazenia zmien je nasledovný:

1. Zvolíme si commit, v ktorom chceme zmeny zobraziť
2. Stlačením tlačidla vedľa mena používateľa otvoríme kontextové menu
3. Stlačením tlačidla „Display changed files (commitID)“ sa zobrazí okno so zmenenými súbormi a zvýraznenými zmenami.

Zobrazenie zmenených súborov

V systéme je možne zobraziť strom so zvýraznenými zmenenými súbormi vo zvolenom commite. Postup zobrazenia je nasledovný:

1. Zvolíme si commit, v ktorom chceme zmeny zobraziť
2. Stlačením tlačidla vedľa mena používateľa otvoríme kontextové menu
3. Stlačením tlačidla „Display files“ sa zobrazí okno so stromom súborov s vyznačenými zmenami v súbore
4. Následne je možné vykonávať rovnaké úkony ako v zobrazení „File system“

21 PRÍLOHA - SYSTÉMOVÁ PRÍRUČKA

Systém CodeReview nemá žiadne špeciálne požiadavky na systém, na ktorom beží, keďže je vo forme webovej aplikácie.

Základná požiadavka:

- Počítač / tablet
- Moderný webový prehliadač
- Pripojenie na internet

Testovanie a vývoj prebiehali vždy v najnovších verziách týchto prehliadačov:

- Mozilla Firefox
- Google Chrome
- Opera

Pri použití najnovšej verzie prehliadača nebude mať používateľ problémy pri používaní nášho systému CodeReview.