

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	4
1.1	ŠTRUKTÚRA DOKUMENTU	4
1.2	ZADANIE PROJEKTU.....	4
2	STANOVENIE CIEĽOV	5
3	PRED PRVÝM ŠPRINTOM	6
4	SEMIENKO	7
4.1	PRIHLÁSENIE POUŽÍVATEĽA AIS LOGINOM	8
4.2	PRIDANIE INFORMÁCIÍ O TÍME NA STRÁNKU PROJEKTU	9
4.3	URČENIE ADMINISTRÁTORA SYSTÉMU	9
4.4	PRIRADENIE POUŽÍVATEĽA K PROJEKTU	10
4.5	ZOBRAZENIE ZOZNAMU PROJEKTOV POUŽÍVATEĽA	11
4.6	NAČÍTANIE SÚBOROVEJ ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE	12
4.7	NAČÍTANIE AST ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE	12
4.8	PREHLIADANIE STROMOVEJ REPREZENTÁCIE	14
4.9	ZOBRAZENIE ZDROJOVÉHO KÓDU ZVOLENEJ ENTITY Z AST-RCS	15
4.10	ZMENA HESLA POUŽÍVATEĽA	15
4.11	REGISTRÁCIA EMAILU POUŽÍVATEĽA	15
4.12	ZVÝRAZNENIE SYNTAXE ZOBRAZENÉHO KÓDU	16
4.13	ZOBRAZENIE ZOZNAMU ODOVZDANÍ PROJEKTU.....	17
5	KORIENOK	19
5.1	ZOBRAZENIE ZOZNAMU PROJEKTOV POUŽÍVATEĽA Z NOVEJ VERZIE AST-RCS	20
5.2	ROZBAĽOVATEĽNÁ STROMOVÁ ŠTRUKTÚRA	20
5.3	PRIRADENIE POUŽÍVATEĽA K PROJEKTU Z NOVEJ VERZIE AST-RCS	21
5.4	ZOBRAZENIE STROMOVEJ ŠTRUKTÚRY VEDĽA ZOBRAZENIA ZDROJOVÉHO KÓDU.....	21
5.5	ZOBRAZENIE GRAFU ODOVZDANÍ PROJEKTU	22
5.6	ZOBRAZENIE ZOZNAMU ZMIEN VYKONANÝCH VO ZVOLENOM ODOVZDANÍ.....	24
5.7	ZOBRAZENIE ZDROJOVÉHO KÓDU ZMENENEJ ENTITY V ODOVZDANÍ SO ZVÝRAZNENÝMI ZMENAMI.....	25
5.8	NAČÍTANIE HISTÓRIE SÚBOROVEJ ENTITY Z AST-RCS	25
5.9	NAČÍTANIE HISTÓRIE ENTITY ZDROJOVÉHO KÓDU Z AST-RCS	26
5.10	ZOBRAZENIE HISTÓRIE ENTITY VO FORME TABUĽKY	27
5.11	POROVNANIE ZDROJOVÉHO KÓDU DVOCH ZVOLENÝCH VERZIÍ ENTITY Z AST-RCS	27
5.12	NAČÍTANIE SÚBOROVEJ ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE PRE NOVÚ VERZIU AST-RCS	29
5.13	NAČÍTANIE AST ŠTRUKTÚRY PROJEKTU DO STROMOVEJ REPREZENTÁCIE PRE NOVÚ VERZIU AST-RCS.....	29
5.14	ZOBRAZENIE ZDROJOVÉHO KÓDU ZVOLENEJ ENTITY Z NOVEJ VERZIE AST-RCS	29
5.15	ZOBRAZENIE ZOZNAMU ODOVZDANÍ PROJEKTU V NOVEJ VERZII AST-RCS	30
5.16	ÚSPEŠNÉ PRIHLÁSENIE S NEPLATNÝM HESLOM	30
6	STONKA.....	31
6.1	VYTVORENIE VIZUÁLU STRÁNKY	32
6.2	VYTVORENIE KONTEXTOVÉHO MENU PRE POLOŽKY	33
6.3	KONTROLA PRÍSTUPU K DÁTAM	33
6.4	FILTRÁCIA HISTÓRIE ODOVZDANÍ.....	35
6.5	STRÁNKOVANIE HISTÓRIE ODOVZDANÍ	36
6.6	PRIRADENIE PROJEKTOV K POUŽÍVATEĽOVI	37
6.7	PRIRADENIE POUŽÍVATEĽOV K PROJEKTU	39
6.8	URČENIE ADMINISTRÁTOROV SYSTÉMU	39
6.9	DEFINOVANIE AKTÍVNYCH ODKAZOV V KONTEXTOVOM MENU.....	40
6.10	ÚPLNÉ ZOBRAZENIE ČÍSLOVANIA RIADKOV	42

Obsah

6.11	VIZUALIZÁCIA ROZBALENIA UZLA STROMU	42
6.12	PRÁZDNE RIADKY NA ZAČIATKU A KONCI KÓDU	43
7	CELKOVÝ POHĽAD NA PROJEKT	44
7.1	TVORBA SOFTVÉROVÉHO PROJEKTU	44
7.2	TVORBA DOKUMENTÁCIE	44
8	VODIČKA	45
8.1	ZOBRAZENIE ZDROJOVÉHO KÓDU SKRÝVA DOKUMENTAČNÉ TAGY	46
8.2	CHÝBAJÚCI OBRÁZOK KONTEXTOVÉHO MENU V STROME	46
8.3	PRI ZOBRAZENÍ POROVNANIA SÚBORU BEZ HISTÓRIE NASTANE CHYBA	46
8.4	PÍSMO V ZOBRAZENÍ CHANGESETOV SA PREKRÝVA	46
8.5	OBNOVENIE SEDENIA PRE ÚDAJE PROJEKTU.....	47
8.6	SÚBEŽNÉ SCOLLOVANIE V POROVNANÍ ZDROJOVÝCH KÓDOV.....	48
8.7	ZVÝRAZNENIE ENTITY V ZDROJOVOM KÓDE	48
8.8	ZBALENIE A ROZBALENIE CELÉHO STROMU	50
8.9	INFRAŠTRUKTÚRA PRE ZAZNAMENÁVANIE UDALOSTÍ.....	50
8.10	ODDELENIE RCS PROJEKTOV OD LOGICKÝCH PROJEKTOV	52
8.11	SPRÁVA LOGICKÉHO PROJEKTU	53
8.12	SPRÁVA POŽIADAVIEK O ZAČLENENIE RCS PROJEKTU K LOGICKÉMU PROJEKTU	54
8.13	SPRÁVA POUŽÍVATEĽOV LOGICKÉHO PROJEKTU	57
8.14	SPRÁVA ALIASOV POUŽÍVATEĽA	59
8.15	NAČÍTANIE ZOZNAMU ZNAČIEK PRIRADENÝCH PREHĽADANEJ SÚBOROVEJ ENTITE	60
8.16	ZOBRAZENIE PROJEKTOV - NASTAVIŤ PEVNÚ ŠÍRKU ZOZNAMOV	62
8.17	PROJEKTY SA ADMINISTRÁTOROVI NAČÍTAJÚ AŽ KEĎ SA PRIHLÁSI DRUHÝKRÁT	62
9	SLNIEČKO.....	63
9.1	KONTROLA PROJEKTOVÝCH PRÁV	64
9.2	ZMAZANIE STARÝCH PROJEKTOV	64
9.3	VYTVORENIE A NAŠTÝLOVANIE SYSTÉMOVÉHO ADMINISTRÁČNÉHO MENU	64
9.4	VÝŠKA RIADKOV PRI ZVÝRAŽŇOVANÍ KÓDU NIE JE URČENÁ SPRÁVNE	65
9.5	DOPLNENIE INFORMÁCIÍ O POŽIADAVKE NA ZAČLENENIE REPOZITÁRA DO PROJEKTU	66
9.6	ZOBRAZENIE KÓDU PO KLIKnutí NA MENO SÚBOROVEJ ALEBO KÓDOVEJ ENTITY V STROME	66
9.7	ZVÝRAZNENIE ENTITY V KÓDE ŽLTÝM OBDĽŽNIKOM	67
9.8	ZLÚČENIE MENNÝCH PRIESTOROV V AST STROME	67
9.9	ZOBRAZENIE TABUĽKY HISTÓRIE ENTITY VEDĽA STROMU	67
9.10	PREMENOVANIE POLOŽIEK COMPARE, HISTORY A GRAPH V KONTEXTOVOM MENU V STROME	68
9.11	KONTEXTOVÉ MENU V AST STROME A HISTÓRII.....	68
9.12	VÝBER IBA DVOCH VERZIÍ ENTÍT NA POROVNANIE	70
9.13	ZACHOVANIE HLAVNÉHO POHĽADU PO ZMENE REPOZITÁRA.....	70
9.14	SYSTÉMOVÁ SPRÁVA PROJEKTOV A REPOZITÁROV	71
9.15	ŠTATISTIKY ZNAČIEK SÚ VŽDY POČÍTANÉ PRE REPOZITÁR WHEREISMYCODE.....	74
9.16	SPRÁVA ALIASOV POUŽÍVATEĽA	75
9.17	ZÁKLADNÉ ŠTATISTIKY NAD PROJEKTOM	76
9.18	USER S ADMINISTRÁTORSKÝMI PRÁVAMI SI MÔŽE ZRUŠIŤ ADMINISTRÁTORSKÉ PRÁVA.....	77
9.19	STRÁNKOVANIE PRI NAČÍTANÍ AST PROJEKTOV DO DATABÁZY.....	77
9.20	FILTRÁCIA HISTÓRIE ODOVZDANÍ - DOPRACOVANIE	77
10	CELKOVÝ POHĽAD NA PROJEKT PO PIATOM ŠPRINTE	79

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 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 funkcionality. Každá iterácia bude obsahovať nové požiadavky na systém, ktoré budú jednotliví členovia tímu postupne riešiť a tím 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 pridelovania projektov administrátorom. Následne budú používatelia systému môcť pracovať s jednotlivými pridelenými projektmi. V projekte bude možné zobrazíť celkovú štruktúru súborov a entít projektu. Systém dokáže zobrazíť rôzne verzie projektu a rozdiely medzi týmito verziami. Pri prehliadke kódov bude následne možné kód komentovať a pridávať značky rôzneho typu a obsahu. Ďalšia funkcionality 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**
- **Piradenie 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**

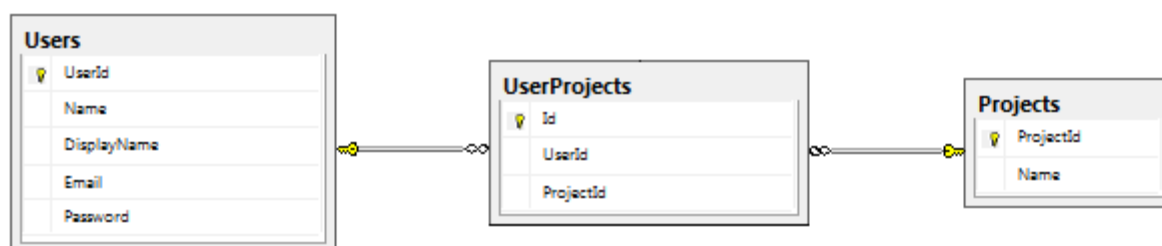
4.1 PRIHLÁSENIE POUŽÍVATEĽA AIS LOGINOM

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 zobrazíť 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

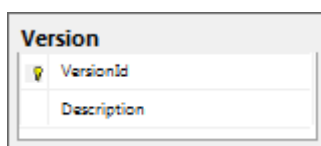
V databáze systému *CodeReview* je možné priradiť používateľovi rolu administrátora, vďaka čomu bude môcť administrátor vykonávať administratívne úkony.

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.

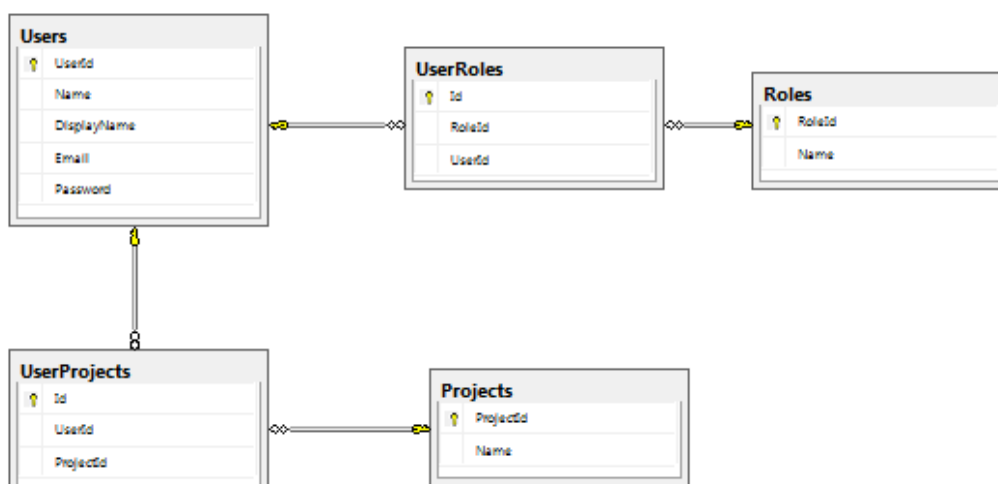
Riešenie

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 verzii.



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 zobrazíť 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
            });
        }
        //Vloženie 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 svojim projektom. Na to, aby mohol používateľ prísť k danému projektu je nevyhnutné mu zobrazíť 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 . Vytvoríť model pre uloženie týchto projektov a zobrazíť ich prehľadne používateľovi.

Riešenie

Na načítanie projektov z databázy je použitá metóda

```

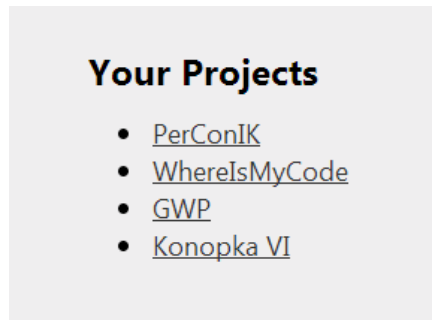
public static List<Project> GetUserProjects(User user)
{
    using (var dc = new CRDataContext())
    {
        return dc.UserProjects.Where(x => x.UserId ==
user.UserId).ToList().Select(x => x.Project).ToList<Project>();
    }
}

```

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 referencií 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 následne 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 zobrazíť 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 nahráť údaje o všetkých kódových entitách daného projektu. Vstupom do tejto funkcie je projekt, ktorého entity chceme načítať. K dispozícii 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 funkcii (*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 ("

"). 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 naštylovať. 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 navrhnuť 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 iní 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 *TreeTypeEnum*. Momentálne existujú dva typy:

- *TreeTypeEnum.CodeEntities* - kódové entity
- *TreeTypeEnum.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

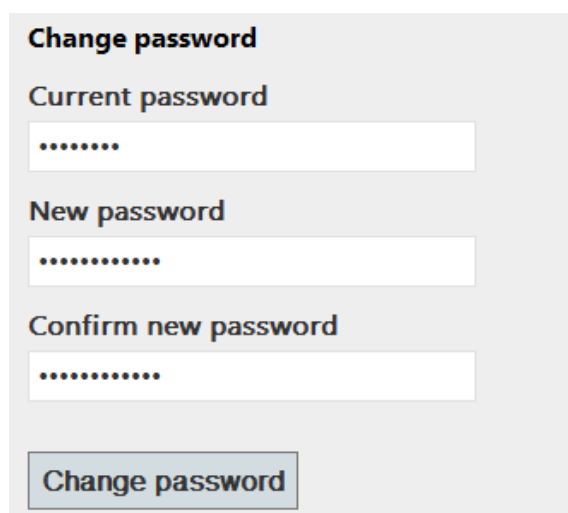
Na zabezpečenie požadovanej funkcionality bola implementovaný kontrolér *FileRequestController*, ktorý slúži na načítanie obsahu zvolenej entity. Tento kontrolér prijíma presný identifikátor verzie (*versionId*) zvolenej entity, na základe ktorej spraví dopyt pre získavanie obsahu entity zo systému AST-RCS. 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

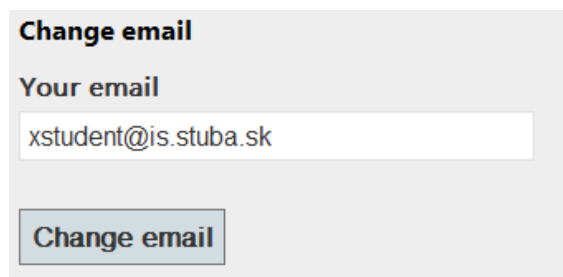
4.11 REGISTRÁCIA EMAILU POUŽÍVATEĽA

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 najpriateľnejšie riešenie pre náš projekt.

Návrh

Po analýze problematiky bol vybraný 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 prehladnutí 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 odovzдания 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 čísel 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 jednoduchšej internetovej stránke.

Riešenie

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

Pri implementácii 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.

```
// vyhľadanie 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 vykonal. Usera (používateľa) ukladáme osobitne z dôvodu, že v informáciách o changesete je iba ID používateľa. Informácie o používatelovi 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 verzioacom 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 verzii 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 ROZBALOVATEĽNÁ 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 rozbaľiť. Pri veľkých projektoch to zabezpečí rýchlejšiu a prehľadnejšiu prácu zo stromovou vizualizáciou.

Analýza

Je potrebné zvážiť, akým spôsobom je možné doceliť 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 použitá 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 zbalovanie a rozbalovanie

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 použitiu 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 <div>. 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 funkcionálnosť 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 odovzdání 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 operácii bol analyzovaný príbeh - **Zobrazenie zoznamu odovzdání 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:

Znázornenie 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:

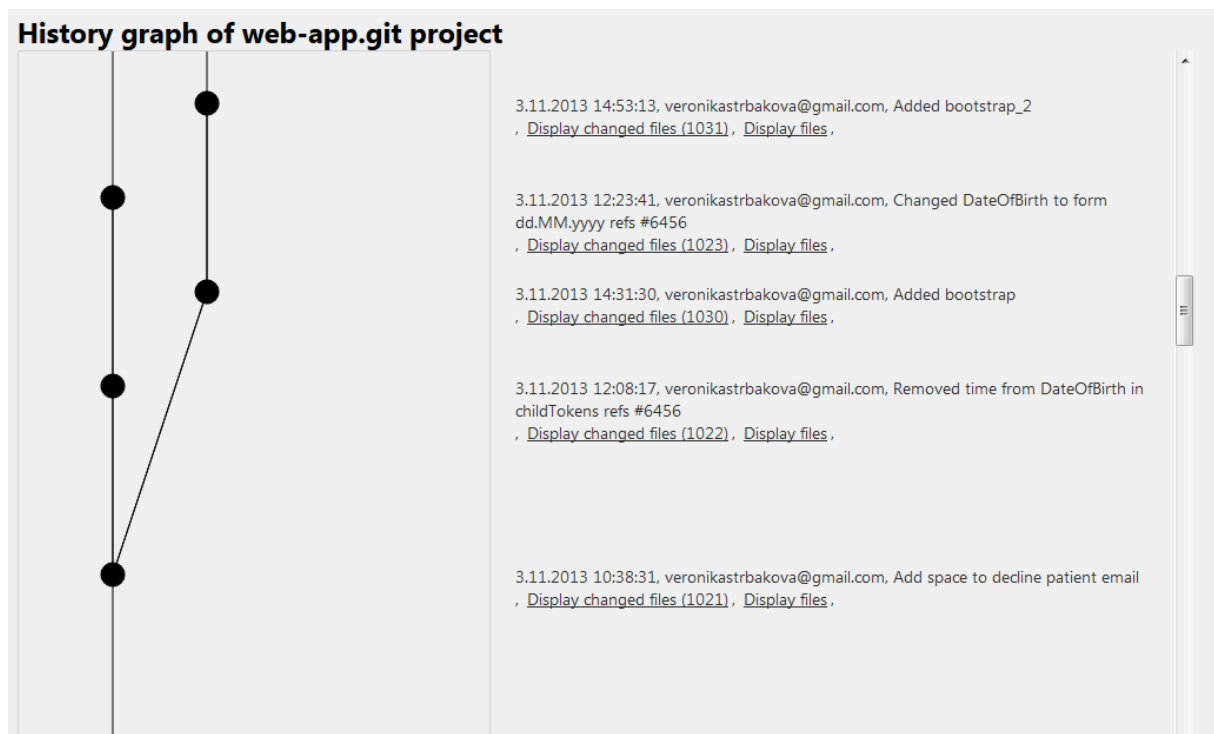
```

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

```

Táto transformácia je zadaná 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 kódach. 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 porovnávanie súborov. Podľa výstupu označíme pridané, zmazané a zmenené riadky v zdrojovom kóde. Označené riadky zobrazieme v používateľskom rozhraní.

Riešenie

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

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

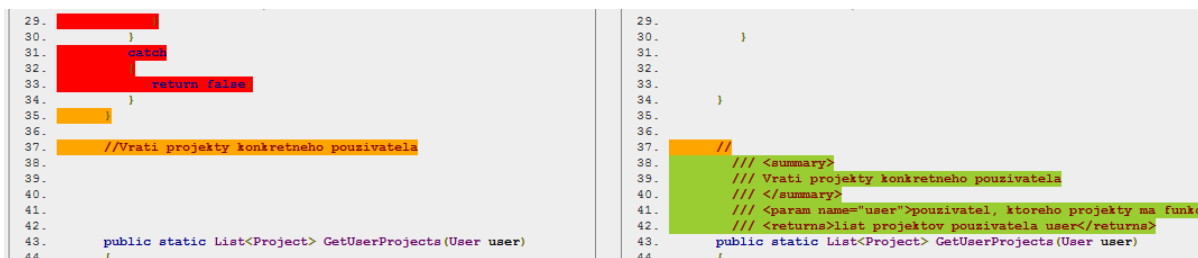
Metóda označuje riadky v kóde, na ktorých nastala zmena. V prípade zmazania súboru alebo vzniku nového súboru sa zmeny nezisťujú.

```
var d = new Differer();

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. 23 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

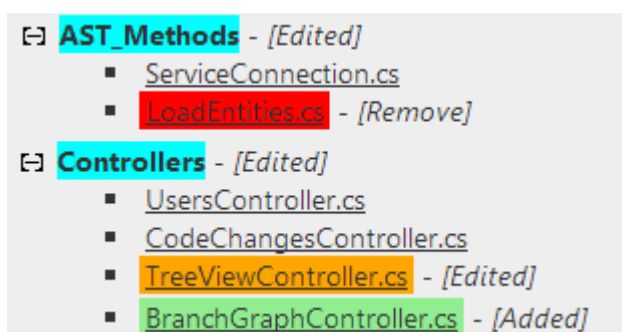
Zobrazenie zdrojového kódu zmenenej entity vo zvolenom odovzdaní. Úlohou bolo graficky zobraziť zmeny v stromovej štruktúre projektu zo zvoleného odovzdania z predchádzajúceho stavu.

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.GetFilesChangesets(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 nejakou 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 ošetrovanie, 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 grafický 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 verziách 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 zvýrazia

```
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).

Versions to compare of
CodeReview/CodeReview.Web/Controllers/HomeController.cs

ID	Author	Timestamp	Check
1353	TFS\xchlebana	14. 11. 2013 21:58:45	<input type="checkbox"/>
1318	TFS\xsamuhel	13. 11. 2013 11:27:06	<input type="checkbox"/>
1263	TFS\xchlebana	12. 11. 2013 10:41:27	<input type="checkbox"/>
1095	TFS\xchlebana	8. 11. 2013 9:02:47	<input type="checkbox"/>
1079	TFS\xsamuhel	7. 11. 2013 21:56:53	<input type="checkbox"/>
274	TFS\xchlebana	30. 10. 2013 16:26:44	<input type="checkbox"/>
273	TFS\xskrisa	30. 10. 2013 15:49:15	<input type="checkbox"/>
271	TFS\xskrisa	30. 10. 2013 13:54:26	<input type="checkbox"/>
220	TFS\xchlebana	3. 10. 2013 12:38:06	<input type="checkbox"/>

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 upravila.

Riešenie

V novej verzii 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 zobrazíť 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 verzii nutné v prvom rade zdefinovať, 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“ slúžia v novej verzii 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 verzii.

Riešenie

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

Testovanie

Testovanie je prevedené rovnako ako v prechádzajúcej verzii.

5.14 ZOBRAZENIE ZDROJOVÉHO KÓDU ZVOLENEJ ENTITY Z NOVEJ VERZIE AST-RCS

Úlohou bola zmeniť už vytvorené funkcionality nášho systému pre prácu s novou verziiu 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 *Committer* 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**

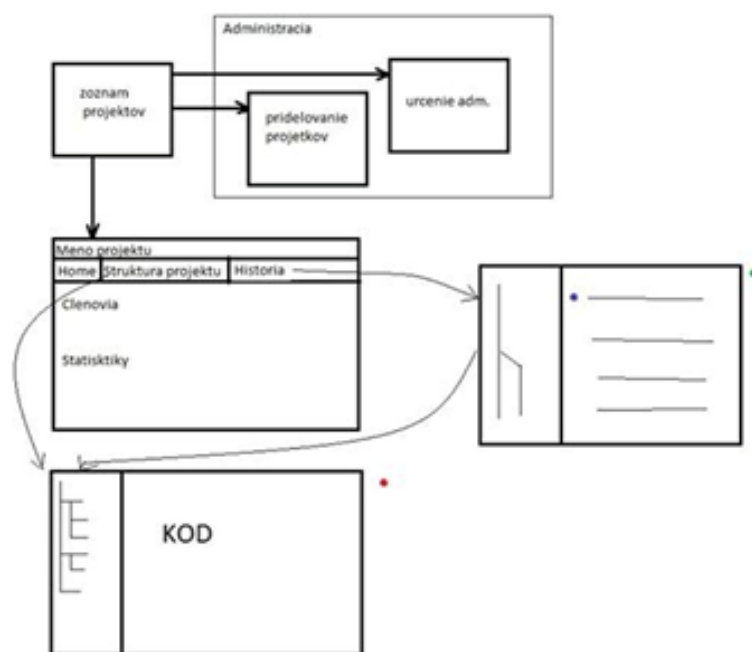
6.1 VYTVORENIE VIZUÁLU STRÁNKY

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.

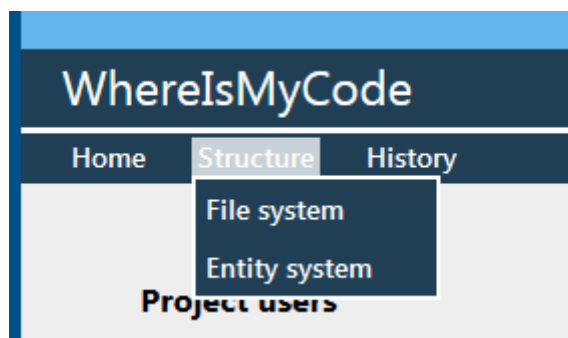
Návrh



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 prídu 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 vyzerieť.

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 situácií 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 zobrazíť 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 píše 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 funkcionálna, ktorá nám zobrazí v grafe odovzdania projektu. Bola vytvorená v druhom šprinte s názvom Korienok, pod príbehom s názvom Zobrazenie grafu odovzdání projektu. Našou úlohou je upraviť túto funkcionálnu 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 odovzdání projektu. To poskytuje výpis všetkých *changesetov* projektu v podobe grafu. Niektoré uzly v grafe však potrebujeme vynechať. Tu využijeme funkcionálnu 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 odovzdání 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 funkcionálnu 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 načítaní 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 verzii 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 možné 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 verzii 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 predošlý 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 verzii príbehu: ZOBRAZENIE GRAFU ODOVZDANÍ PROJEKTU.

6.6 PRIRADENIE PROJEKTOV K POUŽÍVATEĽOVI

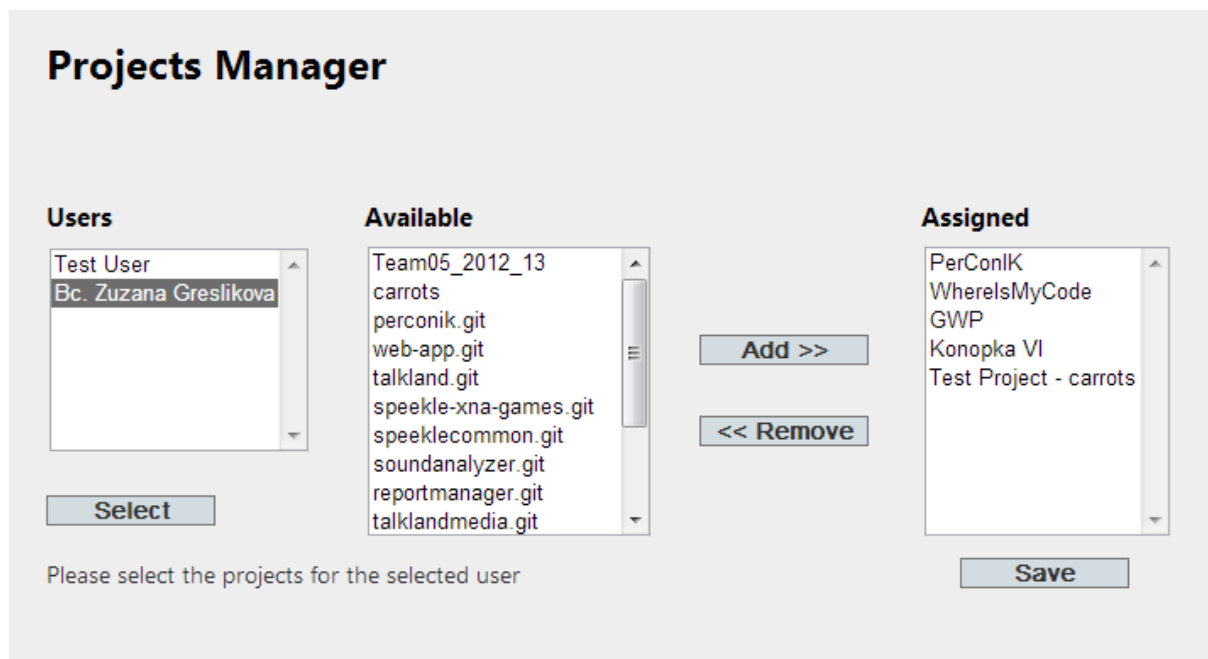
Tento príbeh nadväzuje na príbeh z prvého šprintu 4.4. Niekedy je nevyhnutné pridať jednému používateľovi väčšie množstvo projektov. Je dôležité, aby rozhranie bolo čo najjednoduchšie a aby administrátor nemusel každý projekt priradiť používateľovi individuálne, a aby bolo možné zvoliť skupinu projektov,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

Návrh

Je potrebné vytvoriť rozhranie ktoré umožní multiselekcii 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 multiselekcii 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 administrátorov 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.

The screenshot shows a web interface titled "Admin Manager". It contains a table with two columns: "User name" and "Is Admin". The table lists three users: "Administrator", "test", and "xgreslikova". The "Is Admin" column has checkboxes: checked for "Administrator" and "xgreslikova", and unchecked for "test". A "Submit" button is located at the bottom left of the table area.

User name	Is Admin
Administrator	<input checked="" type="checkbox"/>
test	<input type="checkbox"/>
xgreslikova	<input checked="" type="checkbox"/>

Submit

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:

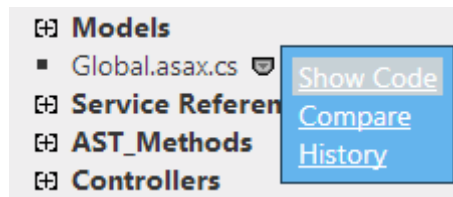
- zobrazenie kódu
- zobrazenie histórie
- porovnanie s predchádzajúcou verziou

Aktívne odkazy pre odovzдания:

- Porovnanie s prechádzajúcou verziou
- Zobrazenie detailov o odovzdaní

Riešenie

Do kontextového 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 rozbaľiť, pri jeho názve bude zobrazený príslušný piktogram a to isté platí aj ak sa uzol rozbaľiť nedá.

Analýza

Je potrebné opäť zvážiť, akým spôsobom je možné doceliť 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 rozbaľený 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 odstránený 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 informácií o projektoch z verziovacieho systému AST-RCS, vytvorenie potrebných dátových štruktúr na reprezentáciu údajov alebo zabezpečenie zobrazovania informácií 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 pričínilo 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 verzou.

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 grafe 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 „perconik.git“, ktorý ma najzložitejšiu štruktúru vetiev z dostupných projektov.

8.5 OBNOVENIE SEDENIA PRE ÚDAJE PROJEKTU

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

Riešenie

Pre zjednodušenie práce boli údaje o session načítané do cookies a následne pri zmazení 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

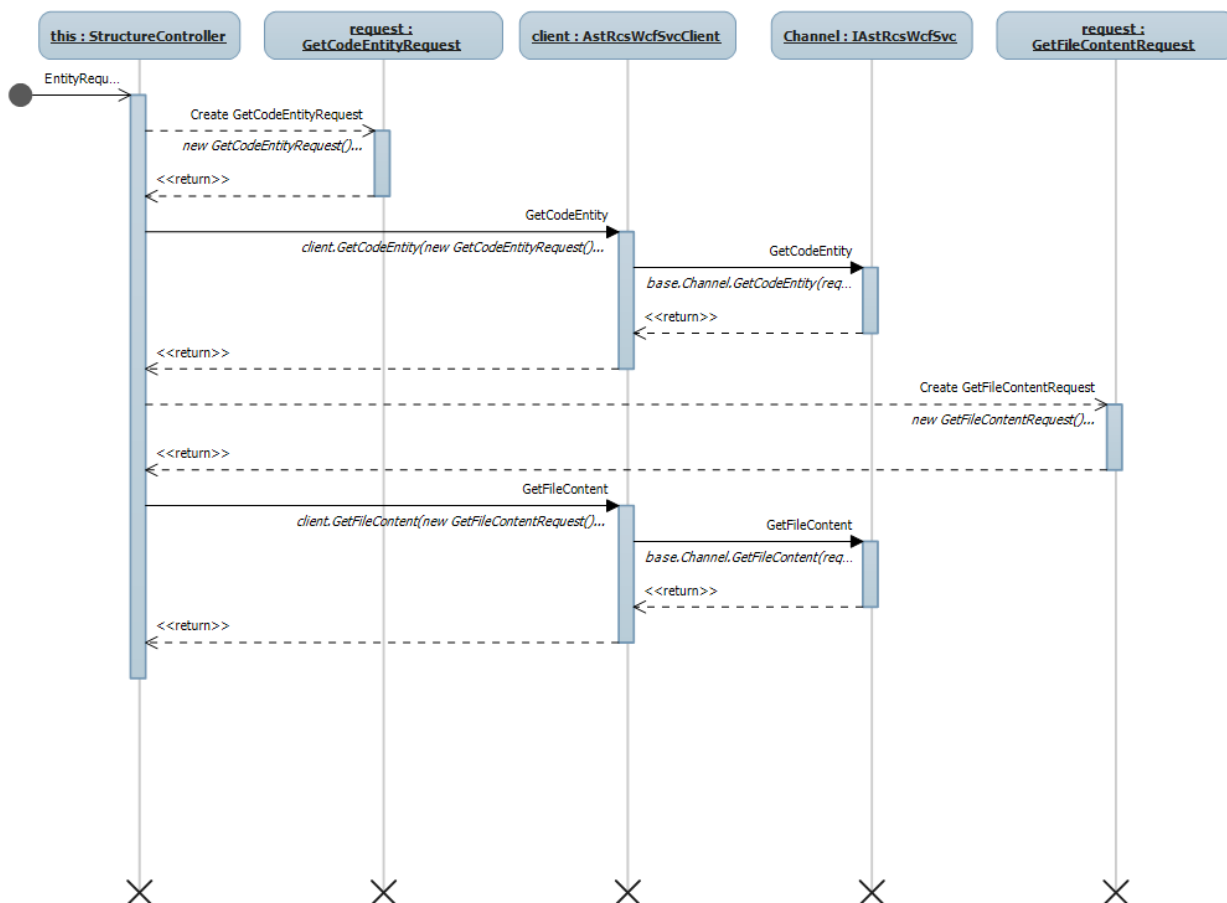
Používateľ chce mať zvýraznené miesto v zdrojovom kóde, ku ktorému je priradená kódová entita, aby vedel túto entitu rýchlo identifikovať bez nutnosti podrobnej analýzy kódu.

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 funkcionálna systém 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 funkcionálna metóda *EntityRequest*. Metóda na základe *versionId* entity získava údaje o umiestnení entity v súbore, a zistí *fileVersionId*. Podľa *fileVersionId* získava obsah súboru, ktorý spolu s 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.

<ul style="list-style-type: none"> [-] CodeReview.Web.Filters <ul style="list-style-type: none"> [-] InitializeSimpleMembershipAttrib <ul style="list-style-type: none"> OnActionExecuting [-] CodeReview.Web.Models [-] CodeReview.Web.Tests.Controllers <ul style="list-style-type: none"> [-] AccountControllerTest [-] CodeChangesControllerTest <ul style="list-style-type: none"> FileEntityHistory CodeEntityHistory [-] HomeControllerTest [-] FileViewerControllerTest [-] BranchGraphControllerTest [-] CodeReview.Web.Properties [-] CodeReview.Web.AstRcsServiceRef [-] CodeReview.Security [-] CodeReview.Web.AST [-] CodeReview.Web.Controllers [-] CodeReview.Web.Enums [-] CodeReview.Web.Database [-] CodeReview.Web.Security [-] CodeReview.Web.ITM [-] CodeReview.Web.Utils 	<pre> 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. using System.Web.Mvc; 7. using Microsoft.VisualStudio.TestTools.UnitTesting; 8. using CodeReview.Web; 9. using CodeReview.Web.Controllers; 10. 11. namespace CodeReview.Web.Tests.Controllers 12. { 13. [TestClass] 14. public class CodeChangesControllerTest 15. { 16. [TestMethod] 17. public void FileEntityHistory () 18. { 19. // Arrange 20. CodeChangesController controller = new CodeChangesController(); 21. 22. // Act 23. ViewResult result = controller.FileEntityHistory() as ViewResult; 24. 25. // Assert 26. Assert.IsNotNull(result); 27. } 28. 29. [TestMethod] 30. public void CodeEntityHistory () 31. { 32. // Arrange 33. CodeChangesController controller = new CodeChangesController(); 34. 35. // Act 36. ViewResult result = controller.CodeEntityHistory() as ViewResult; 37. 38. // Assert 39. Assert.IsNotNull(result); 40. } 41. } 42. } </pre>
--	---

Obr. č. 28 Zvýraznenie entity vo View

Testovanie

Správna funkcionálnosť bola testovaná vizuálne a kontrolou údajov pomocou ladenia.

8.8 ZBALENIE A ROZBALENIE CELÉHO STROMU

Cieľom tohto príbehu je zabezpečiť jednoduchú funkcionálnosť, pri ktorej bude možné stromovú štruktúru zbaľiť resp. rozbaľiť 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 funkcionálnosť, 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 funkcionálnosť

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 ukázkový kód, ktorý sa stará o rozbaľenie stromu.

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

Obr. č. 29 Kompletné rozbaľenie 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

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

- NLog
- Log4Net

Návrh

Pre implementáciu logovania bol vybraný Log4Net pre jeho jednoduchosť v konfiguráciách a implementáciách.

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.Utills.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.Utills.MailClient.Send(String to, String subject, String body) in c:\
\\CodeReview\\CodeReview.Web\\Utills\\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 ODDLENIE 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 priradený 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

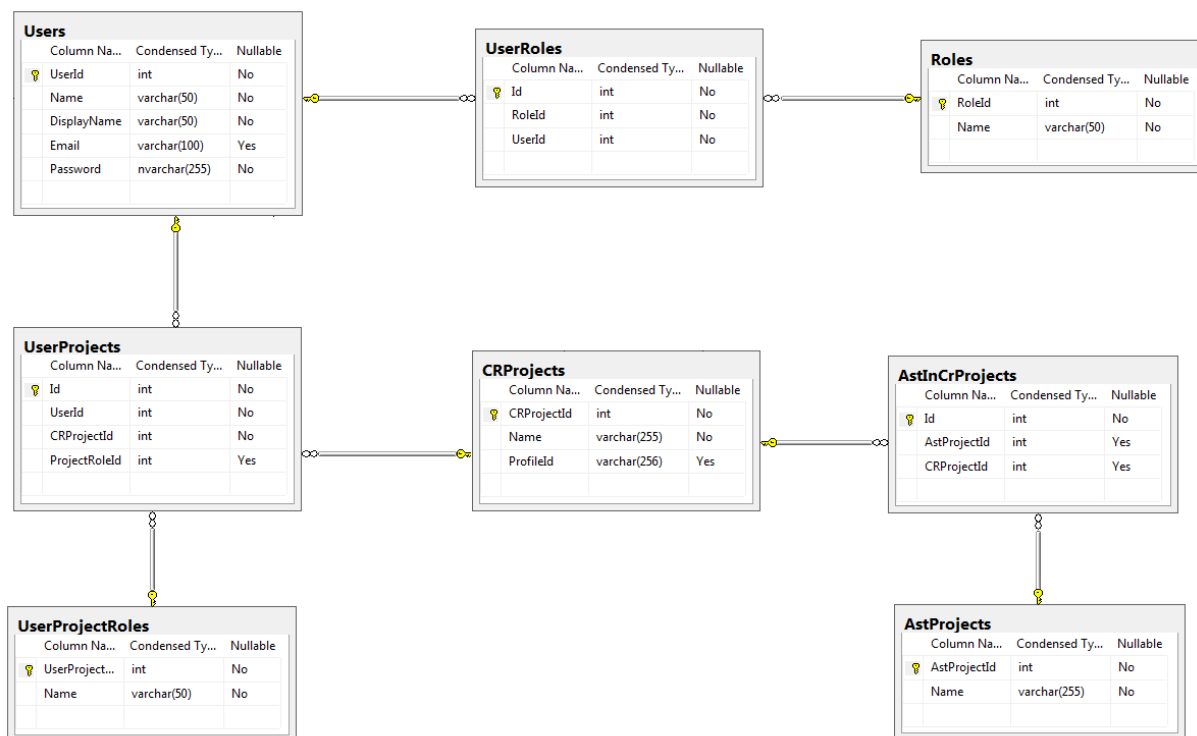
Úloha pozostáva z nasledujúcich úloh:

- Zmena v štruktúre databázy
 - o vytvorenie migračného skriptu
 - o 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.

8 Vodička



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

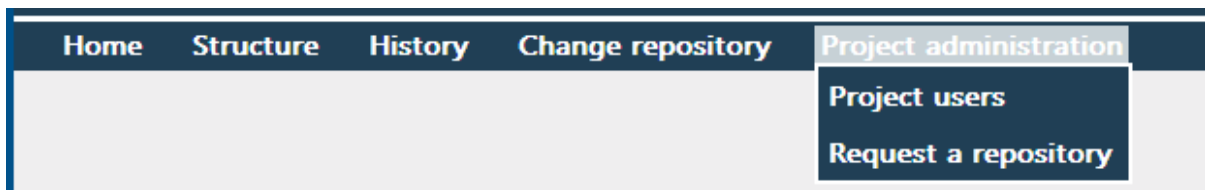
Pri zobrazovaní menu overíme, či je aktuálne prihlásený používateľ administrátor projektu a podľa toho zobrazíme príslušnú položku.

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 požiadany 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.

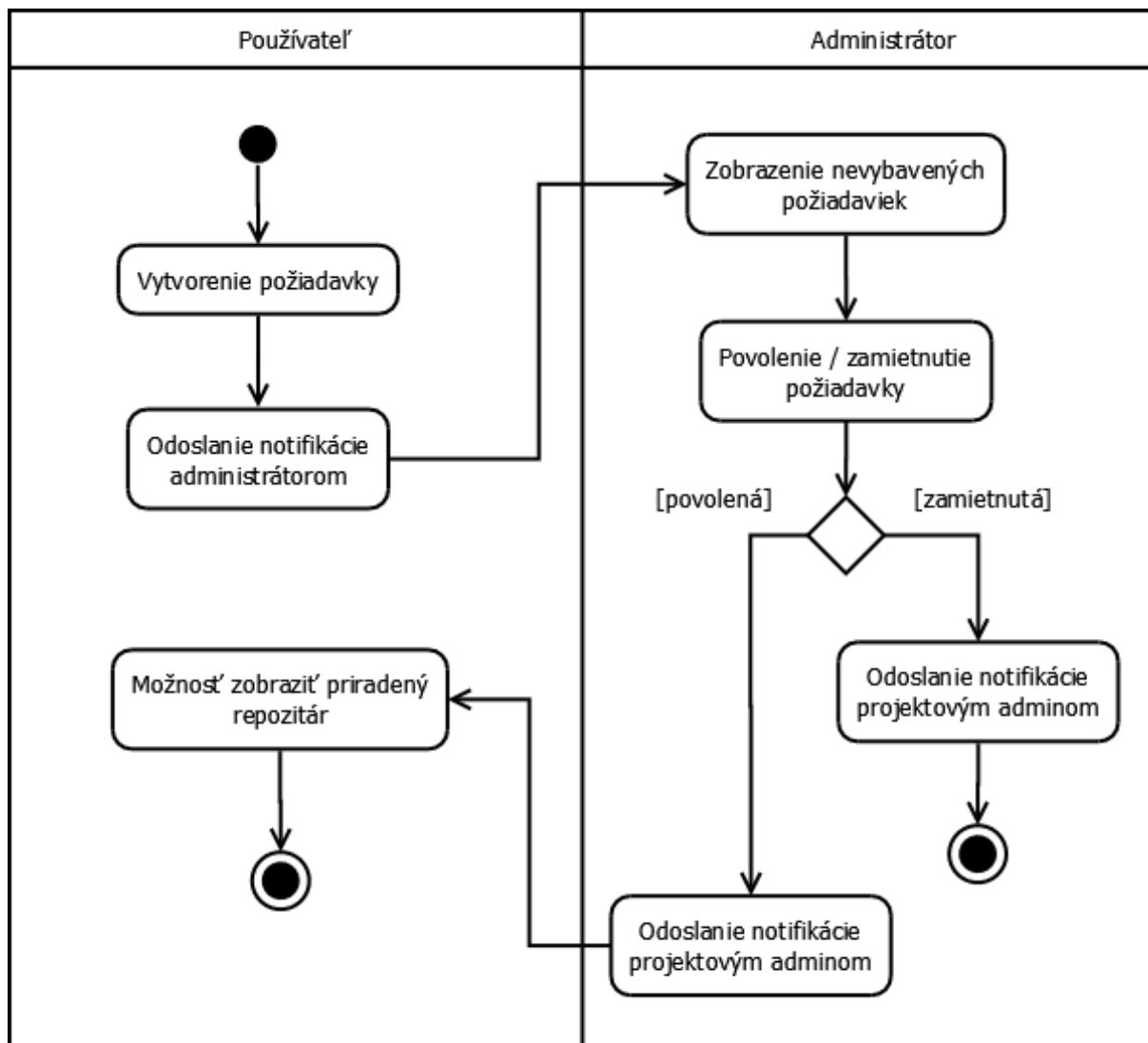
Pridávanie externých repozitárov do systému AST-RCS sa nachádza mimo nášho systému a preto tento problém nebudeme riešiť.

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).



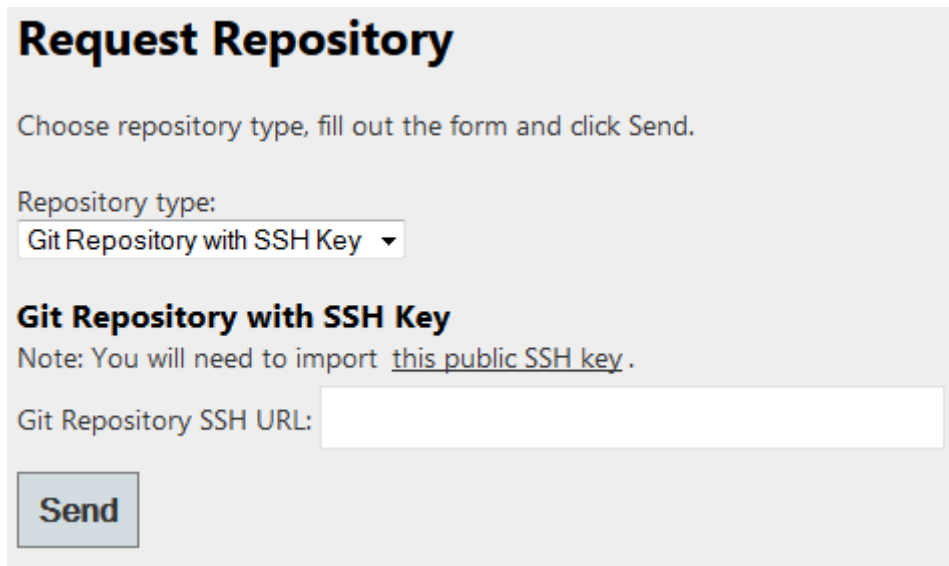
Obr. č. 35 Diagram aktivít pre správu požiadaviek.

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 mu 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 poselať 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.



Request Repository

Choose repository type, fill out the form and click Send.

Repository type:
Git Repository with SSH Key ▾

Git Repository with SSH Key

Note: You will need to import [this public SSH key](#).

Git Repository SSH URL:

Send

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

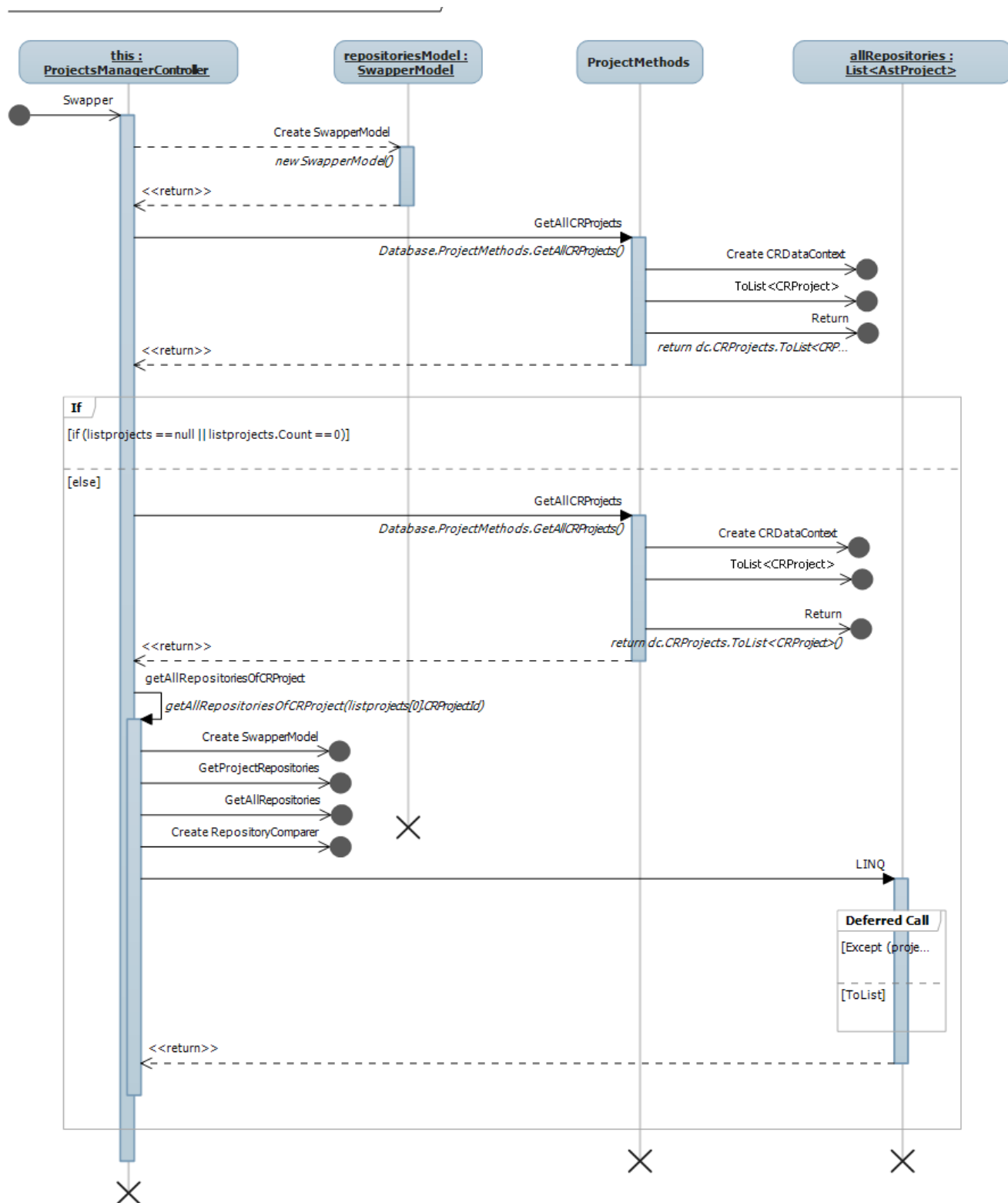
Účastníci: Administrátor projektu

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. Prípád použitia končí.

Návrh

Sekvenčný diagram sa nachádza na obrázku č. 37



Obr. č. 37 Sekvenčný diagram pre vytvorenie modelu

Administrátorovi projektu sú jednotliví používatelia priradení k projektu zobrazení 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

Číslo 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: Prípád prebehol korektno. Zadané zmeny boli uložené do databázy. Následne sa administrátorovi projektu zobrazila potvrdzujúca informácia.

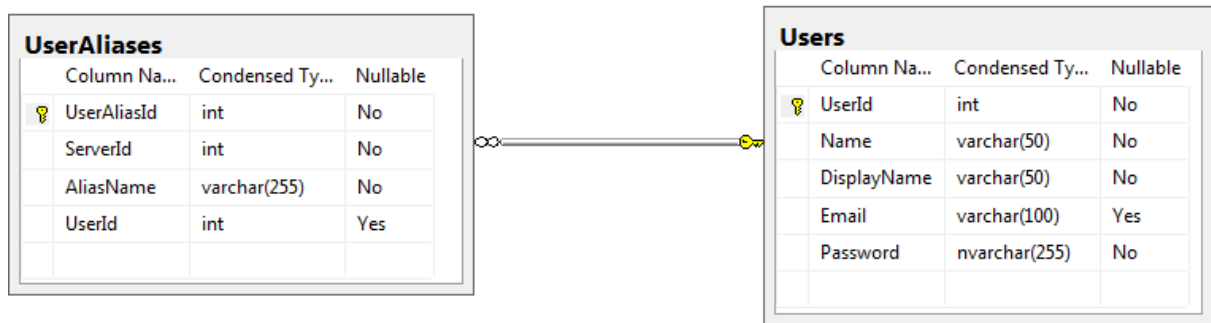
8.14 SPRÁVA ALIASOV POUŽÍVATEĽA

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. Alias 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 kľúča *UserAliasId*
- `GetAliasesOfUser(int UserId)` – funkcia vráti všetky záznamy používateľa, ktorý je identifikovaný podľa *UserId*

V šprinte Vodička príbeh nebol dokončený, neboli implementované časti používateľského 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 Studio 2012.

Našou úlohou je však značky získať a spracovať. Na to použijeme knižnicu *ITMaintenance.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, začiatkový 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 prejdení 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 *ITMaintenance.Driver.dll* do projektu. Po jej pridaní sme implementovali funkciu, ktorá získala list značiek požadovaného súboru. Tú sme naprogramovali v súbore *LoadTags.cs*.

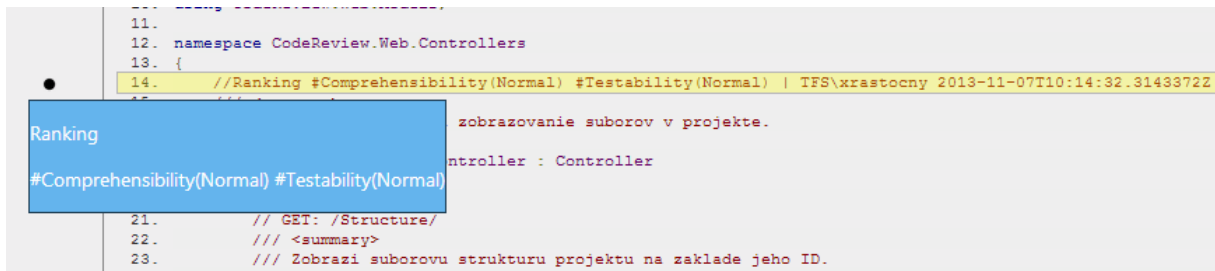
Postup získania značiek je taký ako sme popísali v návrhu. Jeho implementácia teda pozostáva z nasledujúcich častí:

- 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 také ako získanie 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 prejení 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 NAČITAJÚ 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 zobrazíť 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

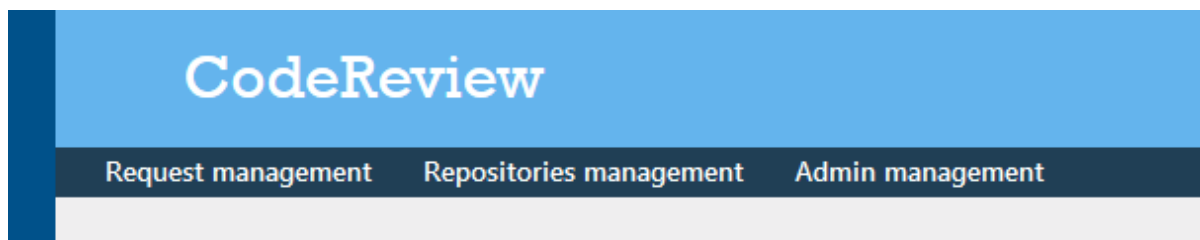
Používateľ vyžaduje jednotné administračné menu systému aby mohol systém pohodlne spravovať.

Analýza

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

- Administrácia požiadaviek
- Administrácia repozitárov
- Správa administrátorov

Riešenie



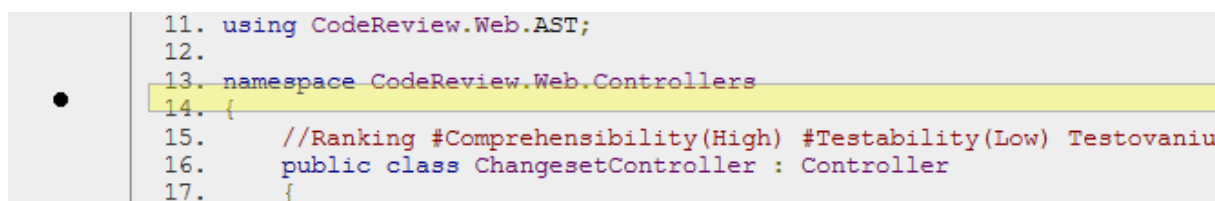
Obr. č. 41 Menu systémovej administrácie

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 implementácii 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 priradenia 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 implementácii, 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 rámkom aby bola zachovaná konzistencia voči zobrazovaniu značiek.

Analýza

V rámci predchádzajúceho šprintu bolo pridané zvýraznenie značiek v zdrojových kódach. 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.cshtml*.

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 históriou 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).

Structure Viewer
WhereIsMyCode
[Expand All](#) / [Collapse All](#)

- [-] CodeReview
 - [-] CodeReview.Web
 - [-] App_Start
 - [-] Properties
 - [-] Filters
 - [-] Models
 - AccountModels.cs
 - CRDatabase.cs
 - DiffModel.cs
 - EntityVersionModel.cs
 - EntityNode.cs
 - EntityHistoryModel.cs
 - BranchGraphModel.cs
 - ChangesetModel.cs
 - ProjectProp.cs
 - TreeViewModel.cs
 - UsersRolesModel.cs

Versions to compare of
CodeReview/CodeReview.Web/Models/AccountModels.cs

ID	Author	Timestamp	Check
236	TFS\χchlebana	20.10.2013 17:02:23	<input type="checkbox"/>
232	TFS\χchlebana	18.10.2013 14:52:36	<input type="checkbox"/>
222	TFS\χchlebana	10.10.2013 13:58:33	<input type="checkbox"/>
220	TFS\χchlebana	3.10.2013 12:38:06	<input type="checkbox"/>

[Compare](#)

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.

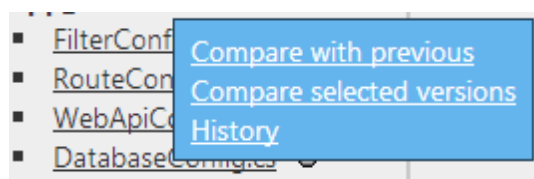
Compare -> Compare with previous version

History -> Compare selected versions

Graph -> History

Riešenie

Boli zmenené názvy aktívnych odkazov oproti predchádzajúcej verzii. Znárodné 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.

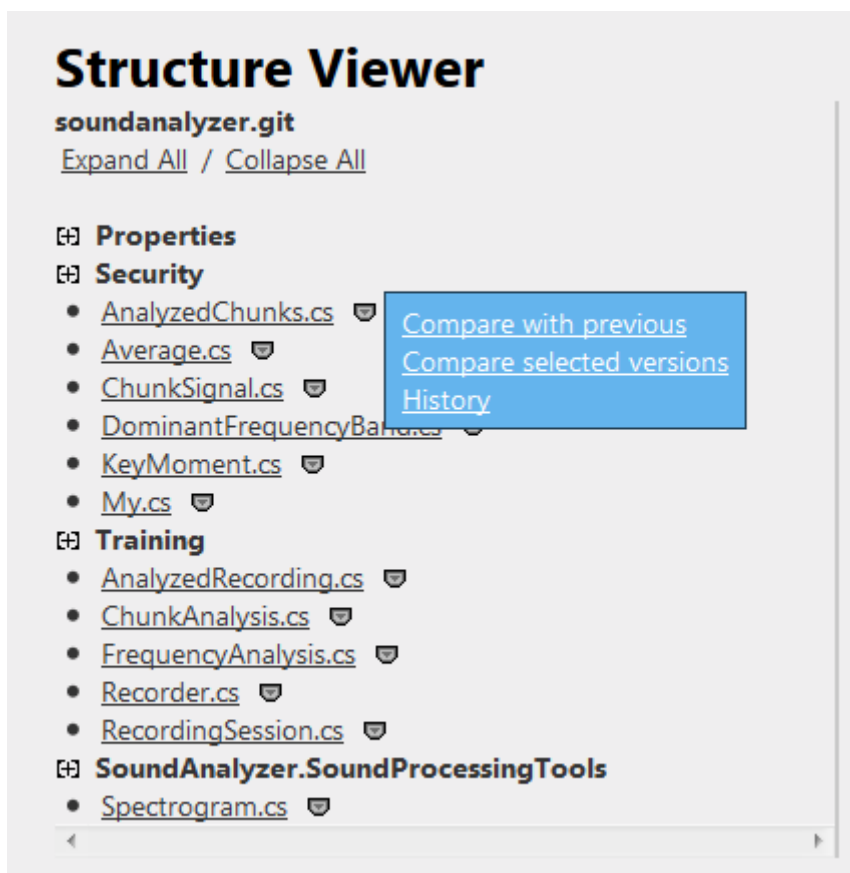
Riešenie

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"></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

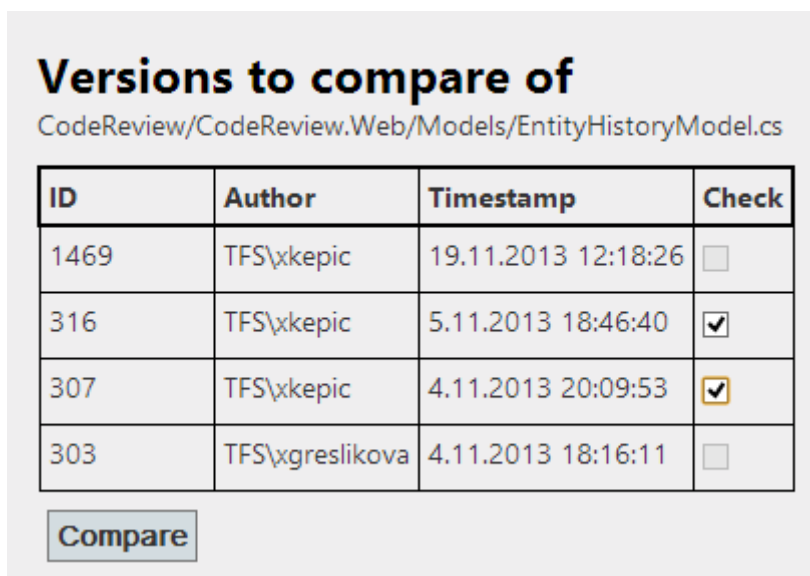
Používateľ požaduje povolenie výberu práve dvoch verzií entity pre porovnanie, aby mal istotu, ktoré dve verzie budú porovnávané, keď označí viac ako dve verzie.

Návrh

V tabuľke s históriou entity doteraz nebolo ošetrované 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.



Versions to compare of
CodeReview/CodeReview.Web/Models/EntityHistoryModel.cs

ID	Author	Timestamp	Check
1469	TFS\ykopic	19.11.2013 12:18:26	<input type="checkbox"/>
316	TFS\ykopic	5.11.2013 18:46:40	<input checked="" type="checkbox"/>
307	TFS\ykopic	4.11.2013 20:09:53	<input checked="" type="checkbox"/>
303	TFS\ygreslikova	4.11.2013 18:16:11	<input type="checkbox"/>

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 nerušené 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 navštívene 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 odovzdání, 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. Prípád použitia končí.

Alternatívne toky:

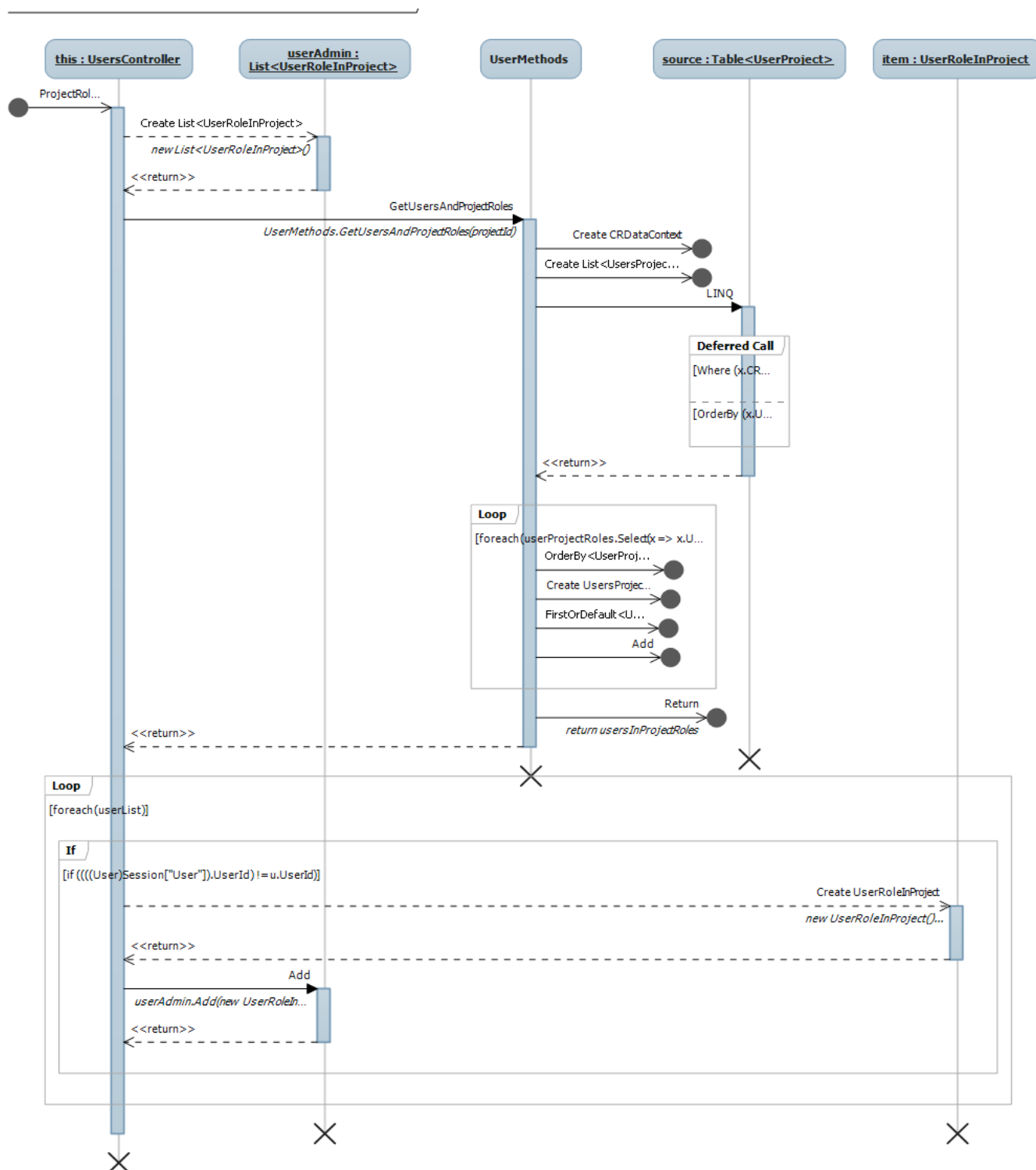
A1 Administrátor zvolí iný projekt

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.

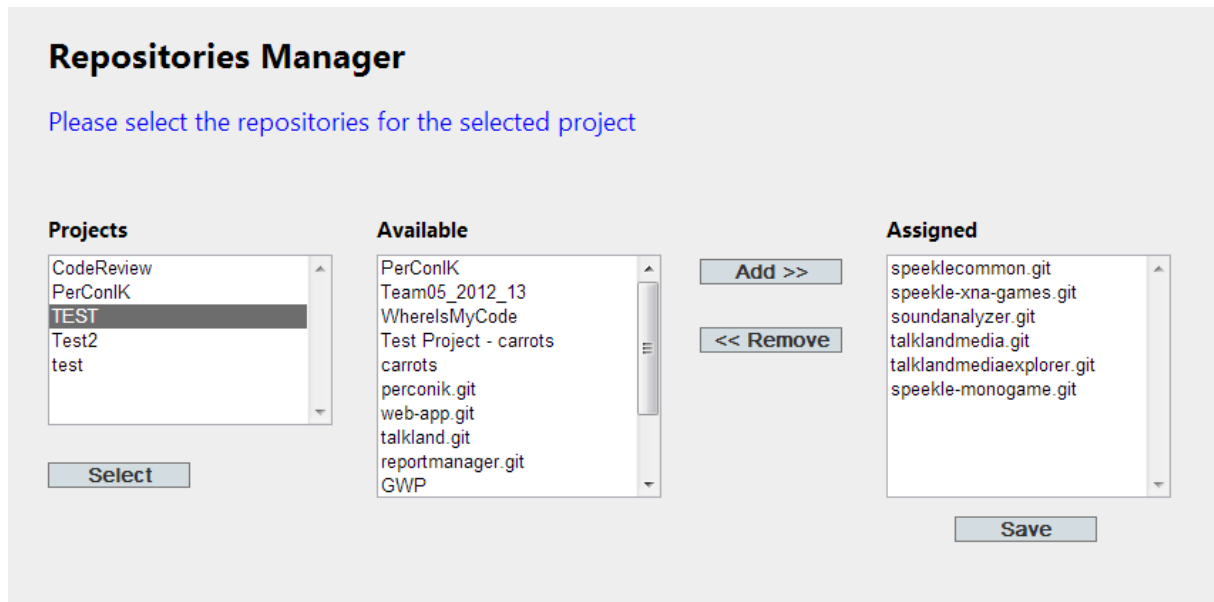
Návrh

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 projekt neobsahuje Po zvolení save je jeho výber uložený.



Obr. č. 52 Zobrazenie správy repozitárov projektov

Riešenie

Číslo 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). Uloženie.

Výstup: Kontrola v databáze. Uloženie 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).

Alias Name	Server	Select
test1	http://mirawen.fiit.stuba.sk:8080/tfs/FIIT	<input type="radio"/>
test2	http://mirawen.fiit.stuba.sk:8080/tfs/FIIT	<input type="radio"/>
test1	https://github.com	<input type="radio"/>
test2	ssh://bitbucket.org	<input type="radio"/>

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

• xscholtz
◦ Number of commits: 14
◦ Last commit: 9. 12. 2013 21:40:40
◦ Number of tags: 8
▪ CodeReview: 1
▪ Recommendation: 4
▪ Todo: 3
• xsamuhel
◦ Number of commits: 6
◦ Last commit: 9. 12. 2013 15:36:33
◦ Number of tags: 5
▪ Recommendation: 1
▪ Todo: 3
▪ Bug: 1

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 naplňaní modelu sa zamedzilo vloženiu aktuálne prihláseného používateľa, a to podmienkou, kde *u* je vkladanie 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

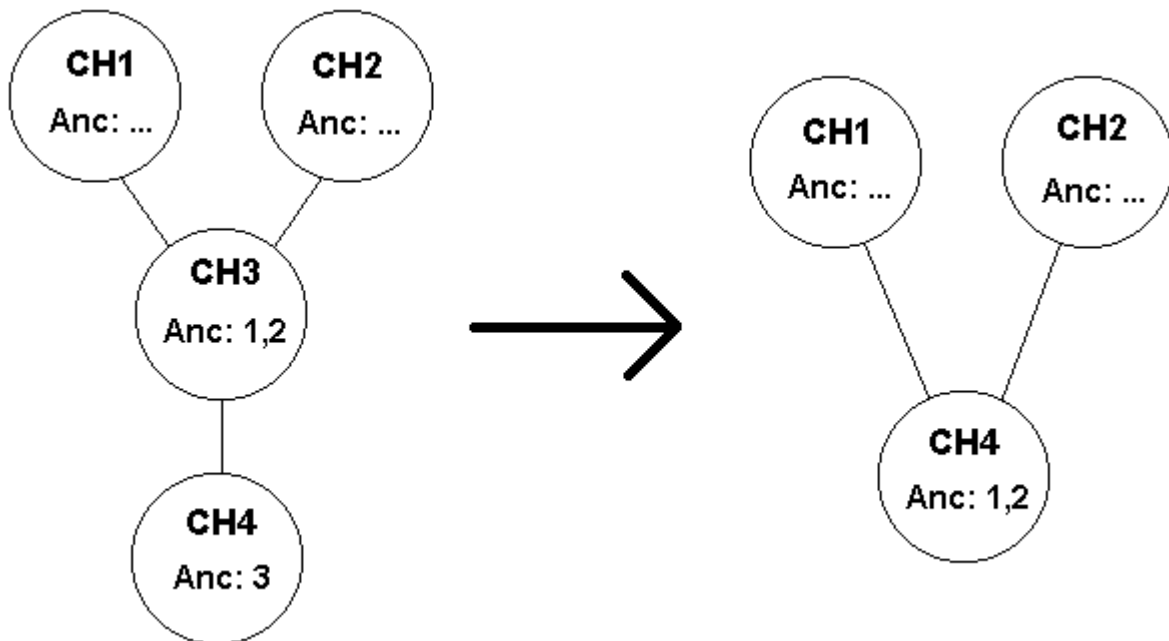
Pre vytvorenie filtra súborovej entity a kódovej entity je postup vykonania filtra rovnaký. Rozdielnym je len spôsob získania changesetov, kedy na nich boli vykonané zmeny. Pomocou ID verzie vyhľadáme changesety, avšak pre súborovú a kódovú entitu zavoláme im príslušne rozdielne funkcie získania changesetov.

Pre úpravu filtra a uloženie predchodcov upravíme mazáciu 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 verzii filtra je len v možnosti 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 verzii 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.

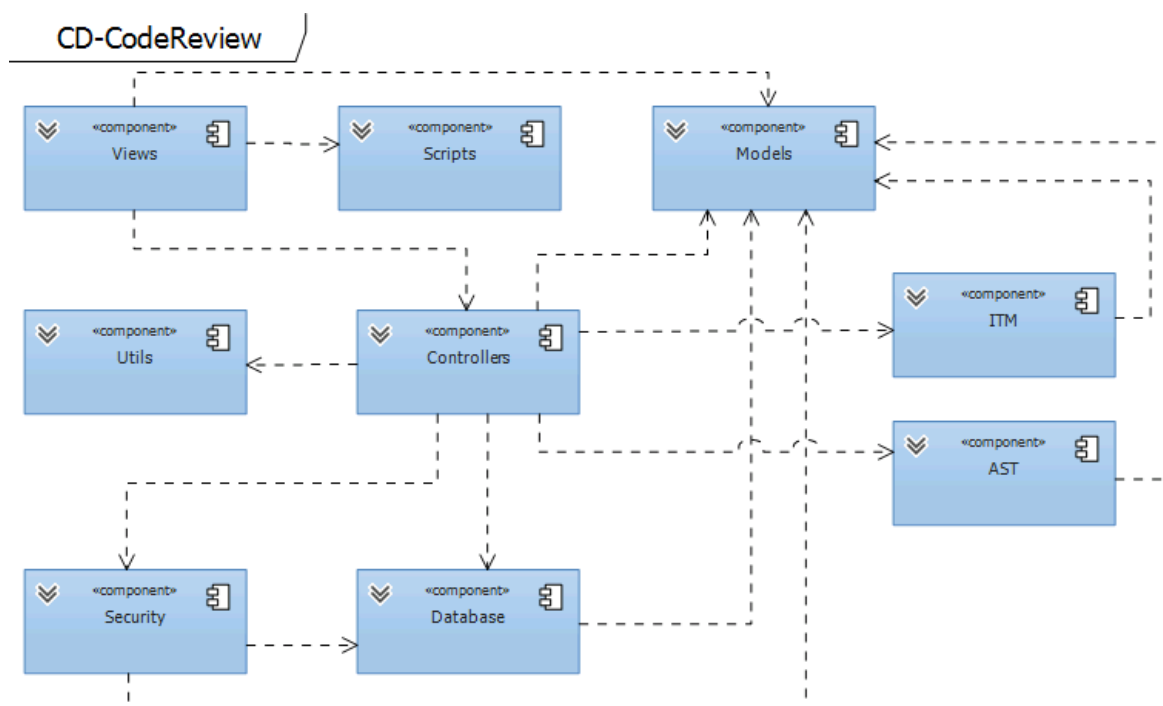
Testovanie

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 aplikácii 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

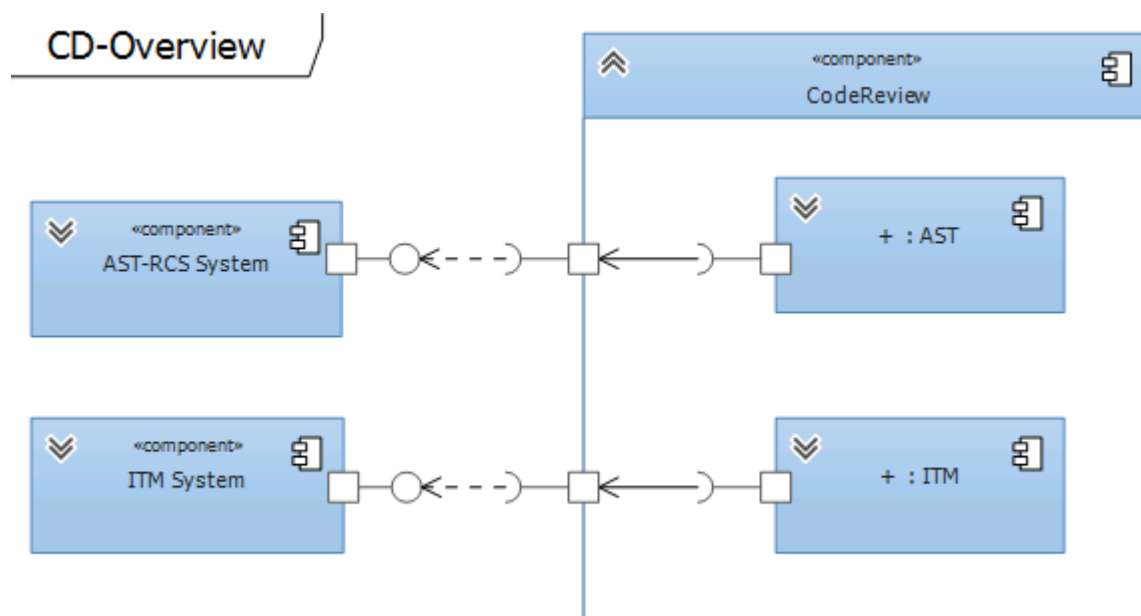
V tejto kapitole je zobrazený pohľad na softvér po 5 šprintoch. Sú tu uvedené „component diagramy“ a „use-case“ diagram systému ako celku. V „component diagramoch“ sú použité iba „<<use>> dependency“ vzťahy. Táto kapitola obsahuje aj dátový model databázy po 5 šprintoch.

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



Obr. č. 57 "Component diagram" CodeReview

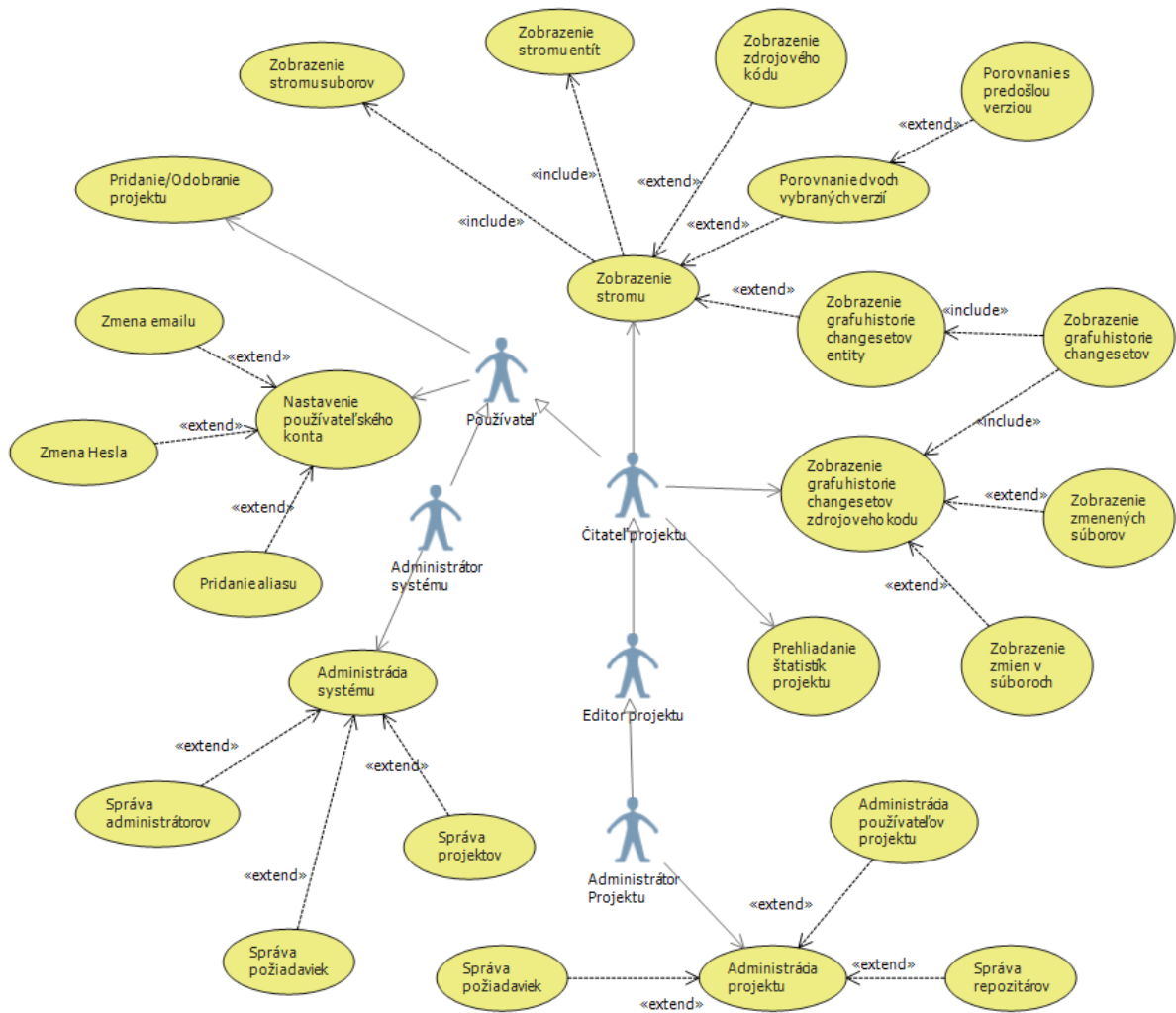
„Component diagram“ zobrazujúci komunikáciu medzi komponentmi CodeReview a inými systémami (ASTRCS a ITM) je zobrazený na obrázku č. 58. Komponenty CodeReview AST a ITM spracúvajú dáta zo systémov ASTRCS a ITM. Dáta sú ďalej posielané do „controllerov“.



Obr. č. 58 Interakcia komponentov CodeReview s inými systémami

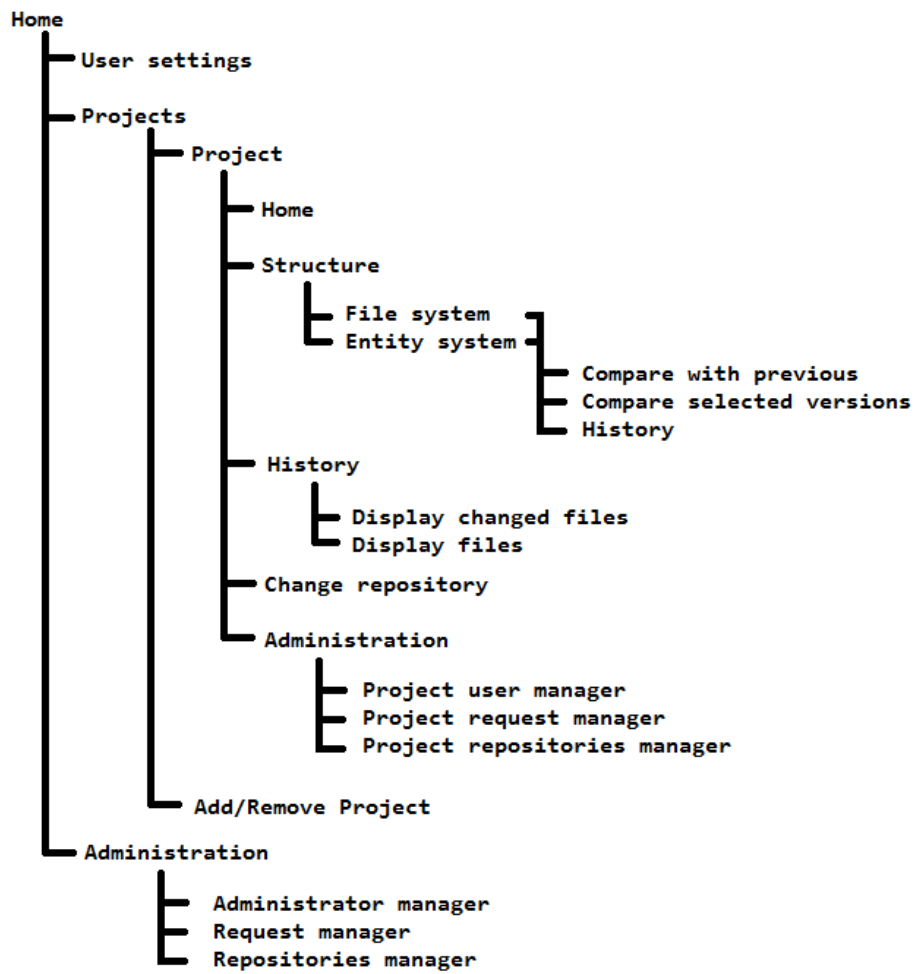
„Use Case diagram“ celého systému je zobrazený na obrázku č. 59.

10 „Big Picture“



Obr. č. 59 Use Case diagram projektu

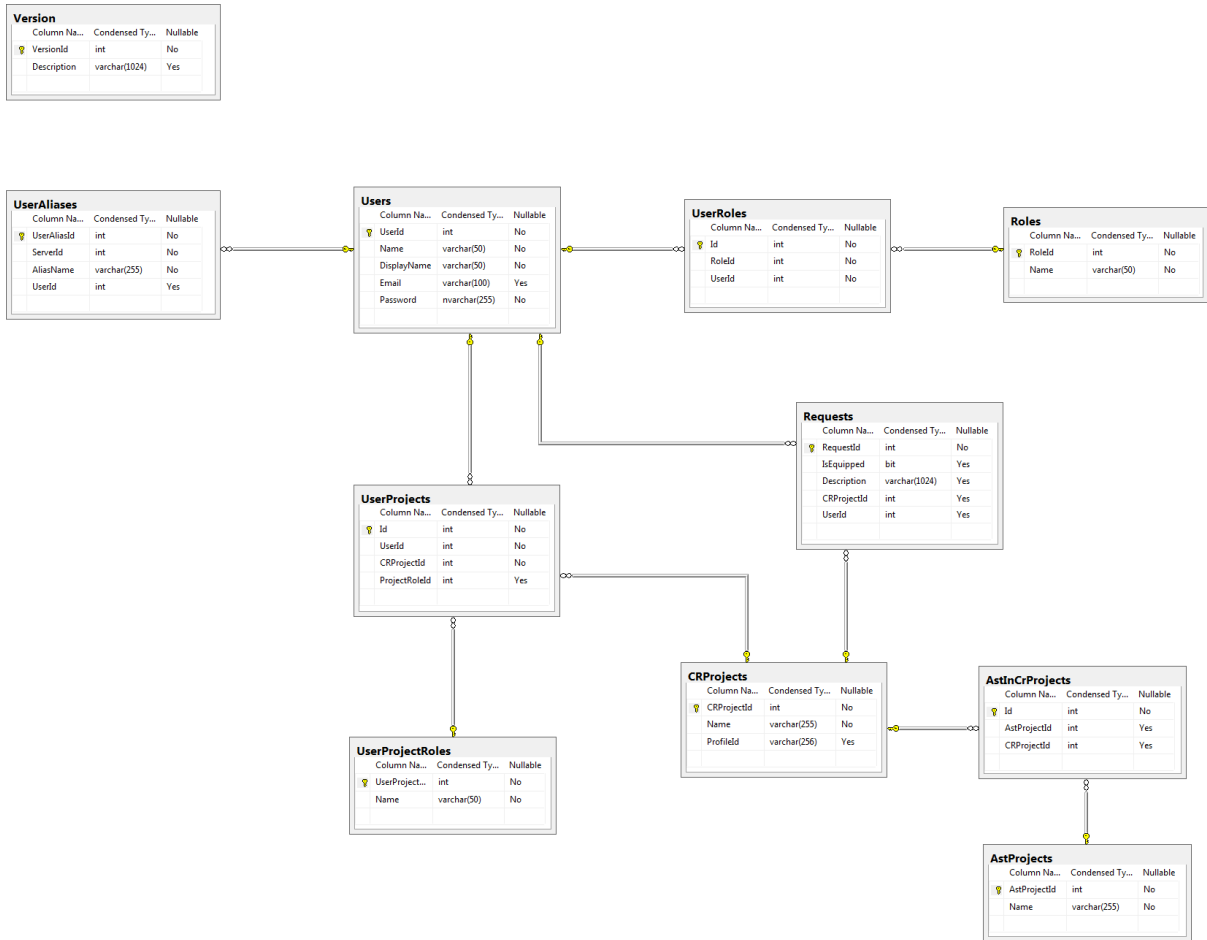
Mapa stránky projektu je zobrazená na obrázku č. 60.



Obr. č. 60 Mapa stránky projektu

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

10 „Big Picture“



Obr. č. 61 Fyzický datový model databázy