Technická dokumentácia User Management modulu

Verzia 13.11.2015

Tabuľka 1. Autori

Autor	Rola
Ivan Beňovic	
Lukáš Hamacek	
Peter Vrana	

Tabuľka 2. História zmien

Verzia	Dátum	Autor	Popis
1.0	13.11.2015	Ivan Beňovic	Vytvorenie dokumentu
1.1	21.3.2016	Lukáš Hamacek	Pridanie sekcie Vyhľadavanie používateľa
1.2	30.3.2016	Peter Vrana	REST služby premiestnené pod API controllery
1.3	17.4.2016	Pavol Beťák	GetUsersByRole služba

Obsah

1 Analýza

Tento modul projektu Administration portal, by mal slúžiť na centrálne nastavenie používateľských práv v rámci všetkých modulov v projekte DevACT.

V projekte DevACTs sa nachádza aktuálne 7 systémov :

- CodeReview
- Cord
- Core
- ITGenerator
- ITMaintance
- Tagging
- UserAcitvity.

Účelom User Management modulu by mala byť, tým pádom správa prístupov ku každému jednému z nich a taktiež možnosť pridania/odobratia v rámci možného budúceho vývoja projektu. Taktiež by mal tento modul vystavovať rozhrania, ktoré budú využívať ostatné systémy na globálnu autentifikáciu /autorizáciu. User Management modul by sa mal zaoberať spravovaním rôznych práv a sprostredkúvať aj služby, ktoré budú pracovať nad databázou používateľov

2 Návrh

Nastavenie práv pre prístup ku systémom by malo byť dostupné iba používateľom s administrátorskými právami. Keďže sme sa dohodli na využívaní NOSQL databázy (Mongo DB) v rámci administračného portálu, implicitne z toho vyplýva, že použitie inej databázy na uloženie dát o prístupoch by nebolo rozumné. Navrhujeme teda vytvoriť kolekcie, do ktorých sa budú ukladať dáta o prístupoch ku repozitáru.

1.1. Návrh štruktúry pre uloženie dát o právach

Navrhujem použiť dve oddelené kolekcie. Jedna kolekcia bude obsahovať zoznam všetkých systémov a druhá kolekcia bude obsahovať zoznam všetkých požívateľov, kde každý používateľ bude mať v sebe zoznam všetkých možných systémových práv.

Kolekcia pre systém by mala obsahovať:

- Názov systému
- Zoznam všetkých možných prístupových práv pre daný systém

Kolekcia pre používateľa by mala obsahovať:

- Meno používateľa
- Heslo (password hash)
- Zoznam dvojičiek [názov systému, právo ku systému]

1.2. Návrh externých rozhraní

Navrhujem použiť rozhranie, ku ktorému sa bude pristupovať pomocou štandardného HTTP protokolu s použitím architektonického štýlu REST, ktoré je dnes veľmi rozšíreným.

1.3. Vyhľadávanie (Filtrovanie) nad používateľmi

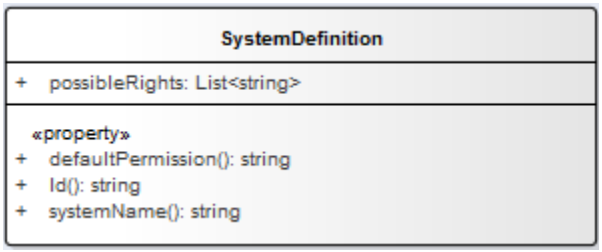
V rámci zadania prišla požiadavka na možnosť vyhľadávať používateľa na základe mena, respektíve jeho prístupu k danému repozitáru. Navrhujeme teda vytvoriť používateľské rozhranie, ktoré umožňuje takéto vyhľadávanie. Avšak pri implementácií je dôležité zachovať flexibilitu riešenia zo zameraním na rozširovanie možných ďalších filtrov.

3 Implementácia

Ako prvé sme implementovali back-endovú stránku aplikácie a to vytvorením a modifikáciami Mongo kolekcií. Identifikovali sme 3 všeobecné práva pre daný systém:

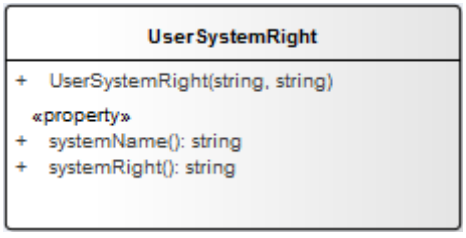
- read
- write
- denied.

Definíciu štruktúry systémového práva môžeme vidieť na obrázku č.1.



Obrázok 1 Štruktúra pre kolekciu systémových práv.

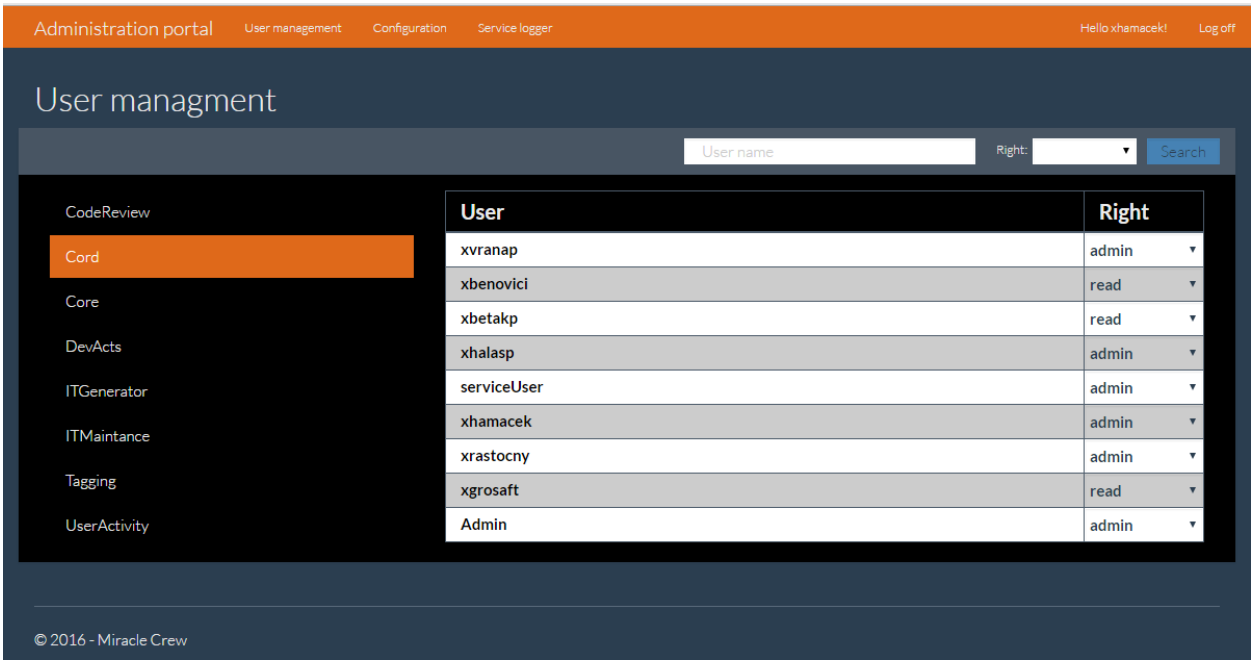
Keďže v našej aplikácií používame O/R mapovač Entity Framework pre správu používateľov, namiesto vytvorenia samostatnej kolekcie pre používateľov sme museli rozšírili už existujúcu kolekciu o zoznam systémových práv. Toto sme dosiahli pridaním poľa systemRights do kolekcie users. Jednalo sa o pole objektov, kde každý objekt obsahuje názov repozitára a názov systémového práva, kde názvy zodpovedajú názvom systémov z kolekcie pre systémové práva. Túto štruktúru môžeme vidieť na obrázku č. 2



Obrázok 2 Štruktúra objektu v poli práv používateľa.

Ako ďalšou úlohou bolo vytvorenie DB API, ktoré bude slúžiť na vyhľadávanie nad používateľskými právami v databáze. Na prepojenie MongoDB sme použili najnovší C# Mongo Driver 2.0.1, pomocou, ktorého sme vytvorili dopyty.

Následne sme sa pustili do implementácií vizuálnej stránky User Managementu. Vizuálnu stránku môžete vidieť na obrázku č.3 (samotný návrh stránky prešiel niekoľkými iteráciami).



Obrázok 3 Ukážka vizuálnej stránky User Managementu.

Vykresľované informácie sme získali pomocou implementovaného DB API pre používateľské práva, ktoré sme následne zahrnuli do modelovej časti MVC architektúry.

Zmena práva pre používateľa sa spustí po kliknutí nového práva pre daného používateľa. Toto kliknutie spôsobí zavolanie Java Scriptovej funkcie, ktorá pomocou AJAX volania zavolá handler metódu pre update používateľského práva pre používateľa. Po úspešnom aktualizovaní práva sa zobrazí na krátky čas zelený infobox s názvom SAVED, v opačnom prípade sa zobrazí chybová hláška.

4.1. Implementácia REST rozhrania pre autentifikáciu

Vstupom do služby na autentifikáciu je AuthenticationRequest objekt, ktorý obsahuje meno používateľa a jeho heslo. Služba počúva na POST metóde, keďže sa odosiela heslo používateľa.

Jej výstupom je objekt AuthenticationResponse serializovaný ako JSON objekt s dvoma atribútmi:

- Authenticated (bool)
- ErrorMessage (string)

V atribúte authenticated sa nachádza výsledok autentifikácie (true/false), v prípade neúspechu je v atribúte errorMessage uvedený dôvod zlyhania autentifikácie. Fungovanie je znázornené na obrázku č.4. Službu je možné volať prostredníctvom endpointu DevActs + suffix (api/users/authenticate).

4.1. Implementácia REST rozhrania pre autorizáciu

Vstupom do služby na autorizáciu je objekt AuthorizationRequest, ktorý obsahuje meno používateľa a názov systému. Služba počúva na Post metóde. Jej výstupom je objekt AuthorizationResponse serializovaný ako JSON objekt s dvoma atribútmi:

- Role (string)
- ErrorMessage (string)

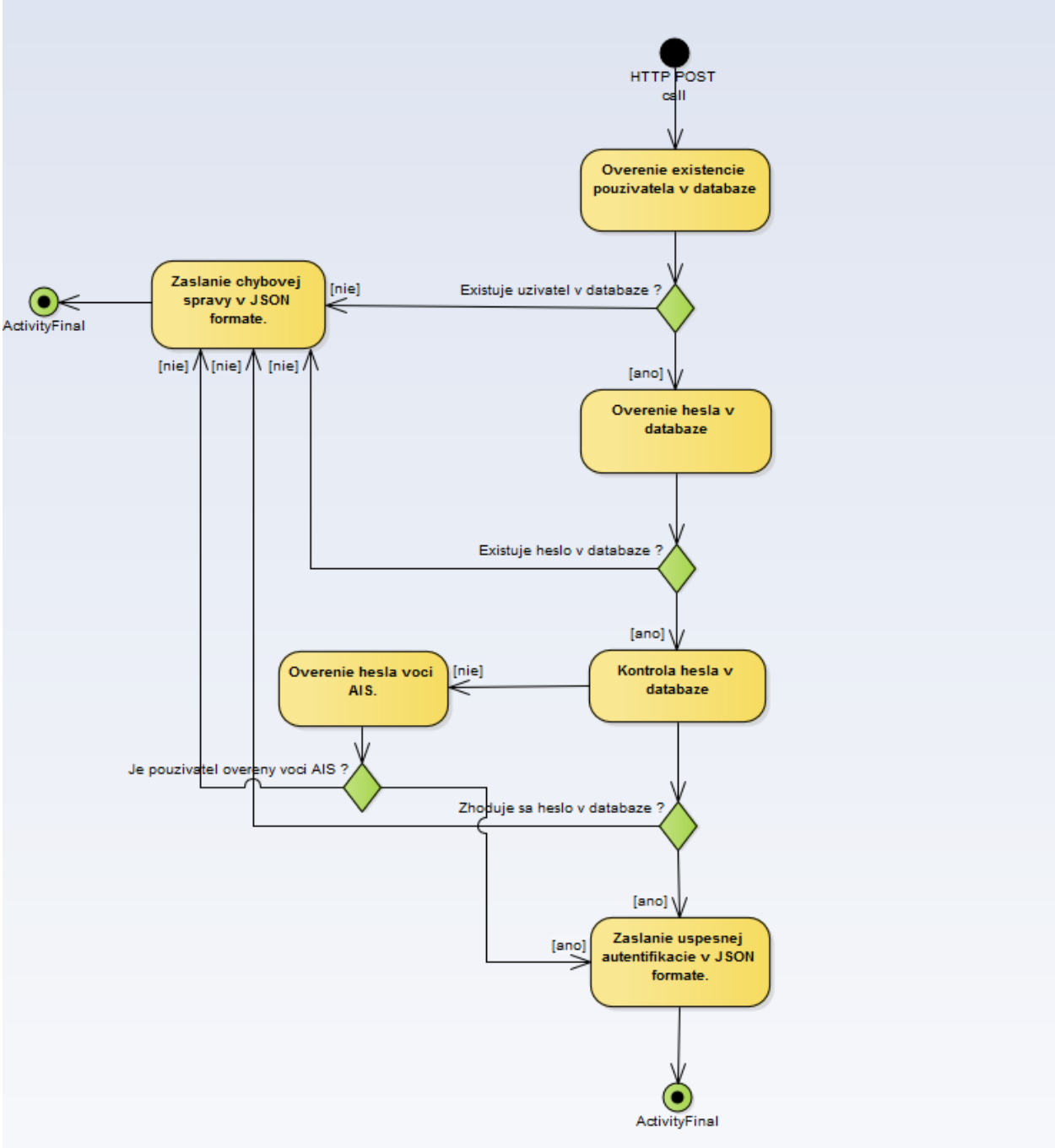
V atribúte Role sa nachádza právo používateľa k danému modulu. V prípade, že niekto zavolá službu pred úspešnou autentifikáciou a názov používateľa alebo názov systému sa nenachádzajú v databáze na výstupe je objekt s atribútom ErrorMessage s popisom chyby. Službu je možné volať prostredníctvom endpointu DevActs + suffix (api/users/authorize).

4.2. Implementácia REST rozhrania pre funkciu GetUsersByRole

Vstupom do služby getUsersByRole sú dva parametre:

- repositoryName – meno repozitára
- role – právo, ktorým by mali používateľ disponovať

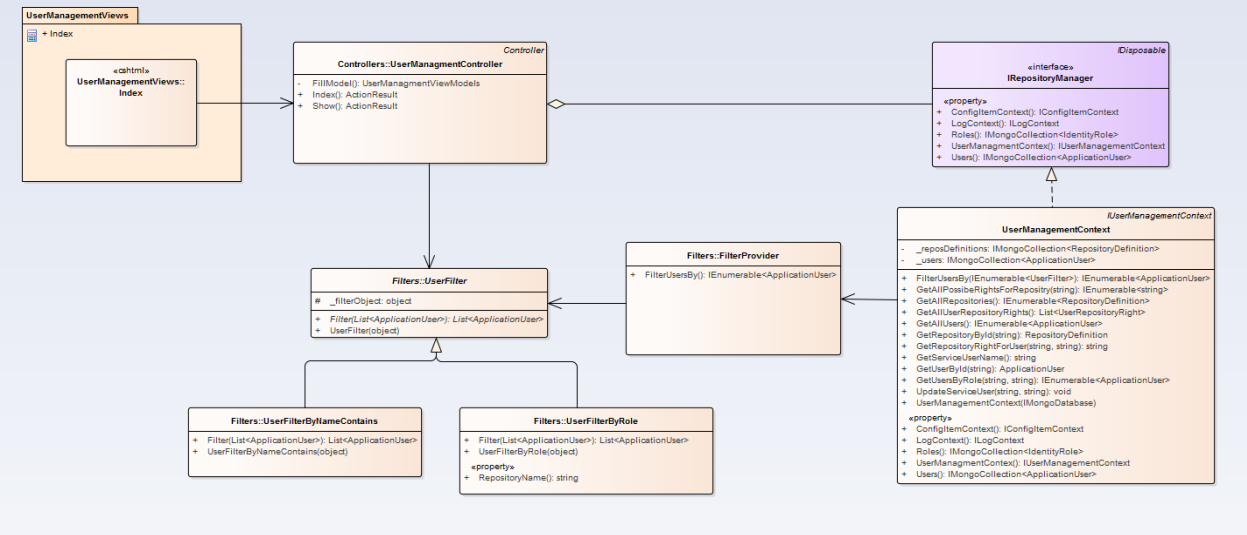
Výstupom tejto služby sú je zoznam používateľov, ktorý disponujú daným právom, pre daný modul. V prípade, že do vstupného parametru „role“, sme uložili hodnotu „all“, služba nám vráti všetkých používateľov. Služba počúva na GET metóde. Službu je možné volať prostredníctvom endpointu DevActs + suffix (api/users/getByRole)



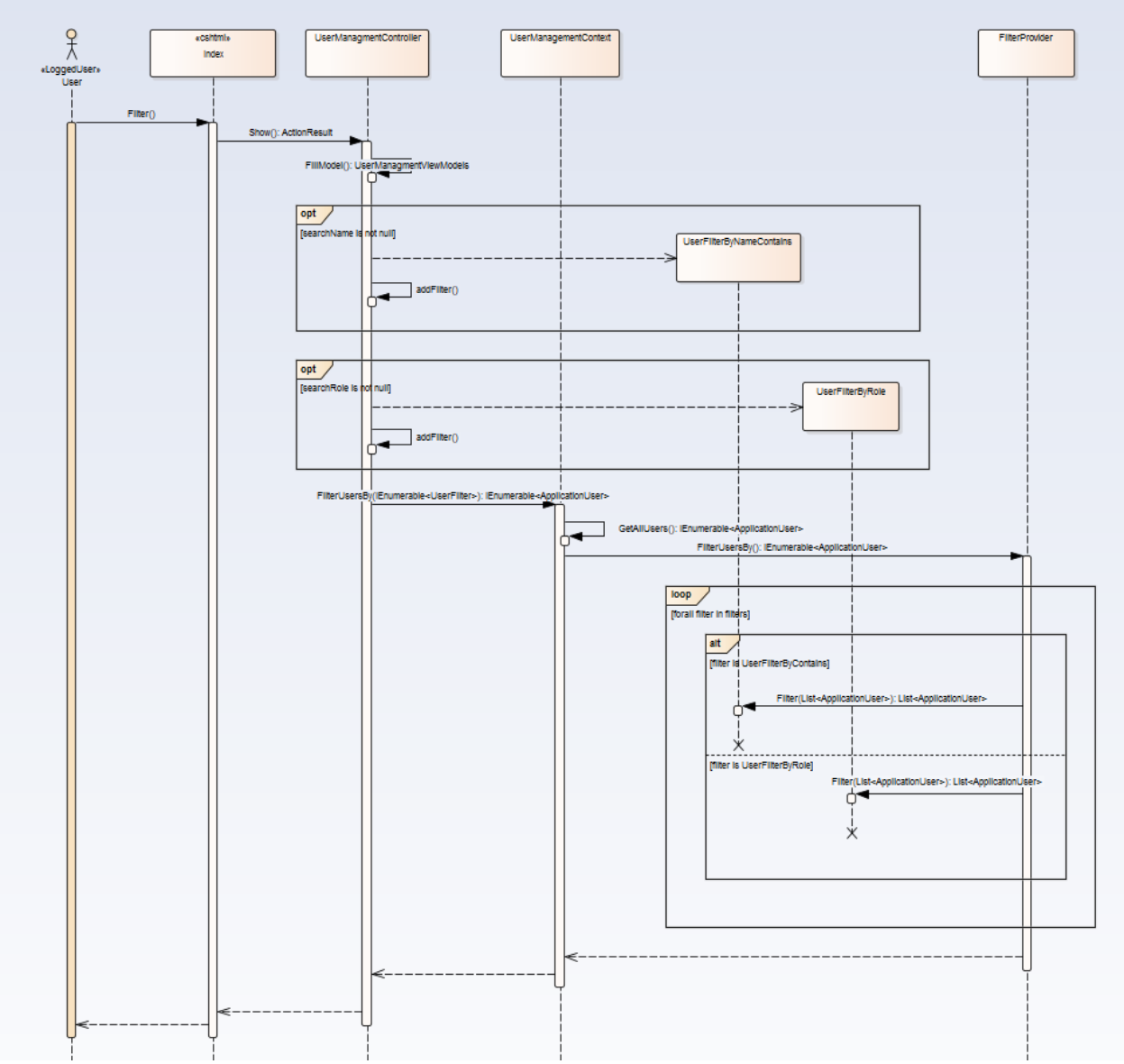
Obrázok 4 Activtity diagram pre autentifikačnú službu.

4.3. Vytvorenie vyhľadávania (filtrov nad používateľmi)

Implementácia filtrov je zobrazená na nasledujúcom diagrame tried.



Interakciu medzi jednotlivými triedami znázorňuje sekvenčný diagram nižšie.


«LoggedUser»
User
«cshtml»
Index
UserManagmentController
UserManagementContext
FilterProvider
Filter()
Show(): ActionResult
FillModel(): UserManagmentViewModels
opt
[searchName is not null]
UserFilterByNameContains
addFilter()
opt
[searchRole is not null]
UserFilterByRole
addFilter()
FilterUsersBy(IEnumerable<UserFilter>): IEnumerable<ApplicationUser>
GetAllUsers(): IEnumerable<ApplicationUser>
FilterUsersBy(): IEnumerable<ApplicationUser>
loop
[forall filter in filters]
alt
[filter is UserFilterByContains]
Filter(List<ApplicationUser>): List<ApplicationUser>
[filter is UserFilterByRole]
Filter(List<ApplicationUser>): List<ApplicationUser>

4 Testovanie

Boli napísané testy pre aktualizáciu používateľských práv ako aj pre obidve externé REST služby. Taktiež boli napísané testy pre vyhľadávacie filtre nad používateľmi.