Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt

Story Teller

Projektová dokumentácia – moduly systému

Vedúci projektu:	Ing. Karol Rástočný, PhD.
Názov tímu:	CoolStoryBro
Členovia tímu:	Bc. Jakub Ondik
	Bc. Patrik Januška
	Bc. Adam Neupauer
	Bc. Martin Olejár
	Bc. Miroslav Hurajt
Kontakt:	storyteller-04@googlegroups.com
Akademický rok:	2016/2017

Obsah

1 Úvod

Tento dokument obsahuje popis všetkých modulov vystupujúcich v systéme. Každý modul obsahuje analýzu, návrh a implementáciu vrátane diagramu tried tam, kde je to vhodné.

2 Modul – Používateľ

Úlohou modulu používateľ má byť riadenie prístupu k jednotlivým častiam a funkciám systému. Zároveň má ponúkať používateľovi možnosť prezentovať sa verejným profilom, obnoviť prístup do systému v prípade zabudnutia alebo straty identifikačných údajov.

2.1 Analýza

Registrácia bude prebiehať zadaním prihlasovacieho mena – emailu a prihlasovacieho hesla, podobne ako prihlásenie. Po registrácii bude používateľovi odoslaný aktivačný a deaktivačný odkaz prostredníctvom emailovej správy. Pomocou týchto odkazov si bude môcť svoj účet aktivovať, čím sa mu umožní prihlásenie, ale aj deaktivovať, čím sa jeho účet zo systému odstráni.

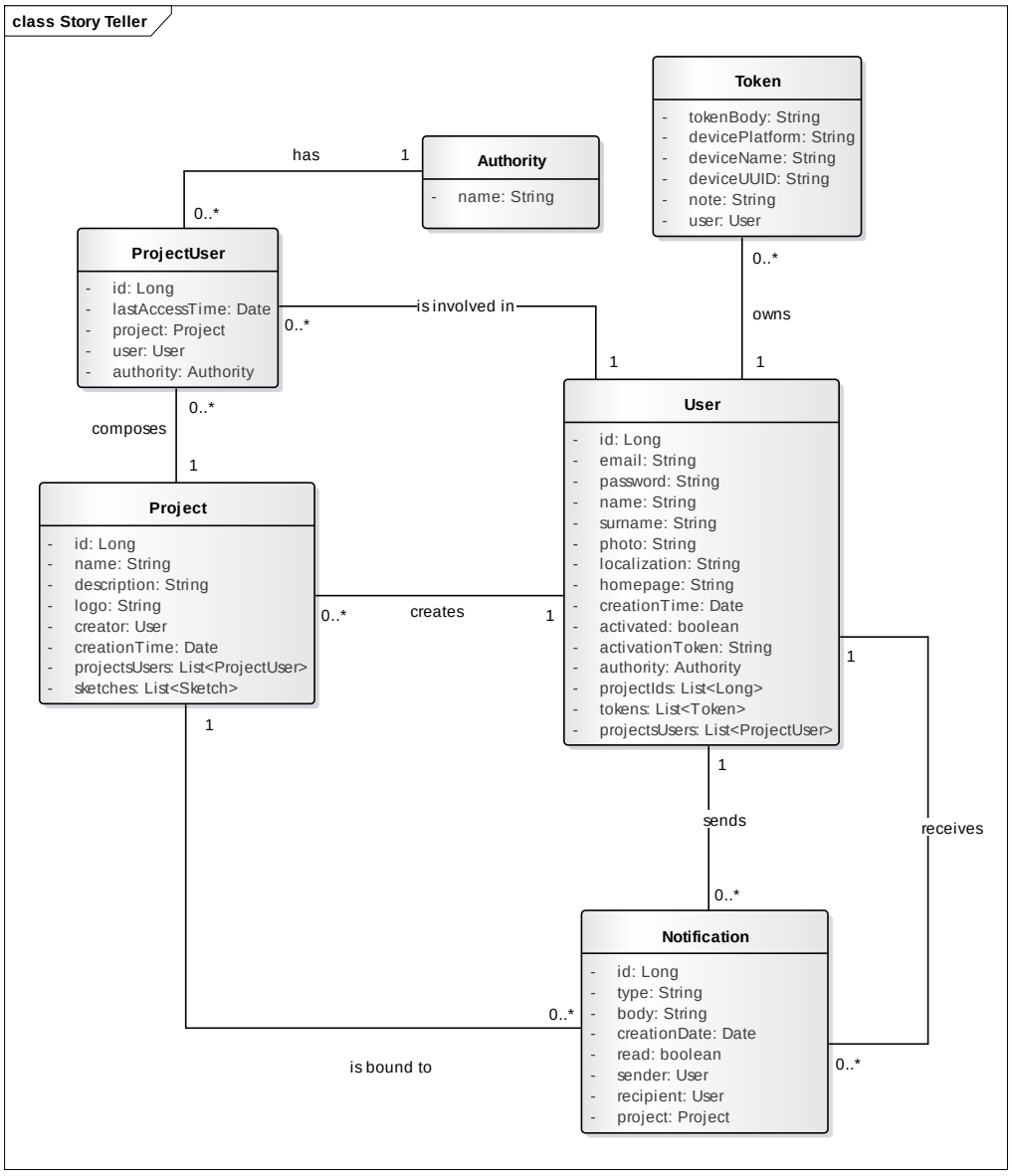
Používateľ bude mať možnosť vyplniť základné informácie o sebe vo svojom profile – meno, priezvisko, homepage a fotografiu a bude mať možnosť zmeniť si heslo. V prípade straty alebo zabudnutia hesla bude mať možnosť odoslania nového hesla na emailovú adresu, pomocou ktorej sa zaregistroval.

Riadenie prístupu bude prebiehať pomocou rolí. Používateľ si bude mať možnosť vybrať hlavnú rolu – rolu analytika alebo zákazníka, v ktorej bude v systéme vystupovať. Používateľ bude mať možnosť túto rolu kedykoľvek upraviť vo svojom profile.

2.2 Návrh a implementácia

Na perzistenciu objektu používateľa reprezentovaného triedou **User** bol použitý OR mapovač s použitím modulu Spring Data rovnako ako na perzistenciu zvyšných tried zobrazených v dátovom modeli na obrázku 1 – **Token** a **Authority**. Trieda **Authority** je určená na riadenie prístupu prostredníctvom rolí, v tomto prípade rolí analytika a zákazníka. Trieda **Token** reprezentuje autentifikačný token pre používateľa – teda nepoužívame HTTP relácie (z angl. sessions). Použitie tokenov zjednodušuje škálovanie aplikácie v budúcnosti, keďže nám umožňuje prechod na bezstavovú (z angl. stateless) autentifikáciu.

K základnému prístupu k údajom v databáze boli použité repozitáre (balík **com.storyteller.repository**) rozhrania, ktoré rozširujú rozhrania modulu Spring Data a umožňujú tvorbu tzv. dopytových metód (z angl. query methods), pomocou ktorých modul Spring Data dokáže vygenerovať dopyty do databázy na základe názvu metódy. Repozitáre taktiež podporujú definíciu dopytov pomocou **@Query** anotácií a JPA syntaxe.



Obrázok 1: Dátový model používateľa

Taktiež boli použité servisy (z angl. services) obsahujúce biznis logiku a riadenie databázových transakcií. K dátam sa teda nepristupuje priamo cez repozitáre, ale cez servisy, ktoré následne v transakciách pristupujú k dátam. Týmto je dodržaný princíp viacvrstvovej architektúry.

V rámci projektu bolo rozpoznaných viacero typov rolí. Ide o tieto roly:

- Zákazník – rola zákazníka umožňuje prezeranie akceptačných testov a skíc a pridávanie ľudí do projektu, resp. prideľovanie ich rolí
- Analytik – rola analytika umožňuje modifikovanie, prezeranie alebo odstránenie akceptačných testov a skíc a vytváranie nového projektu.
- Používateľ – rola charakteristická pre všeobecného používateľa a opisuje spoločné vlastnosti nadtriedy
- Administrátor – rola administrátora určuje špecifické vlastnosti, ktoré môže vykonávať jedine administrátor

2.2.1 Registrácia

Registrácia používateľa je možná prostredníctvom REST endpointu metódou POST, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authentication-controller/createNewUserUsingPOST. Tento endpoint vykonáva kontrolu na duplicitu emailových adries. V prípade správne vyplnených registračných údajov sa vytvorí a uloží do databázy nový objekt triedy **User** s nastavenými atribútmi **email, password, activated, activationToken** a **creationTime**. Na validovanie emailu a hesla boli použité JPA anotácie **@Email** a **@NotNull** s kombináciou anotácie **@Size(min, max)** ošetrujúcou minimálnu a maximálnu dĺžku reťazcov.

Po úspešnom vytvorení nového používateľa je používateľovi zaslaný email prostredníctvom **zoho.com** z adresy **noreply@story-teller.xyz** obsahujúci aktivačný a deaktivačný odkaz. Túto konfiguráciu je možné zmeniť upravením konfiguračného súboru **application.properties** v **/src/main/resources/**. Používateľ si svoj účet môže aktivovať do 12 hodín od odoslania odkazu, inak mu bude účet pri nasledujúcom pokuse o aktiváciu, resp. v pravidelnom nastavenom intervale, zrušený. Deaktivácia účtu môže prebehnúť, pokiaľ účet nie je aktivovaný.

Aktivácia účtu je možná prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authentication-controller/activateUserUsingGET. Na základe atribútu **activationToken** sa vyhľadá používateľ v databáze a zmení sa mu atribút **activated** na true. V prípade neskorej aktivácie účtu je používateľ odstránený z databázy.

Deaktivácia účtu je možná prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authentication-controller/deactivateUserUsingGET. Na základe atribútu **activationToken** sa vyhľadá používateľ v databáze a odstráni sa z databázy.

2.2.2 Prihlásenie

Prihlásenie bolo implementované s využitím konfigurácie modulu Spring Security a servletových filtrov. Tieto filtre zabezpečujú overenie autentifikačného tokenu, ktorý používateľ pri autentifikácii musí priložiť do HTTP hlavičky **X-AUTH-TOKEN** v HTTP požiadavke. Samotná konfigurácia sa nachádza v triede **SecurityConfiguration** v balíku **com.storyteller.config** spolu s triedou reprezentujúcou Spring Security servis používateľa **UserDetailsService**. Po úspešnom prihlásení odoslaním požiadavky metódou **POST** na endpoint **/login** je používateľovi vygenerovaný autentifikačný token reprezentovaný triedou **Token**, ktorý je uložený v databáze. Tento token je v klientskej časti uložený do **local storage** prehliadača, resp. zariadenia a následne pripájaný do HTTP hlavičky **X-AUTH-TOKEN** v HTTP požiadavke pomocou AngularJS interceptora **authInterceptor.js** v **/components/interceptor/**. Pri odhlásení používateľa (REST endpoint **/logout** metódou GET) je tento token odstránený z **local storage** ako aj z databázy.

2.2.3 Úprava údajov v profile

Úprava základných informácií prebieha odoslaním celého objektu používateľa **User** do REST endpointu metódou **PUT**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/user-controller/updateUserProfileUsingPUT. Pre aktuálne prihláseného používateľa sú uložené nové údaje o jeho profile do databázy, konkrétne atribúty **name, surname, photo, localization** a **homepage**.

Získanie aktuálnych dát o prihlásenom používateľovi prebieha odoslaním požiadavky metódou **GET** na REST endpoint, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/user-controller/getUserProfileUsingGET. Pre aktuálne prihláseného používateľa sú vybraté všetky údaje o používateľovi z databázy.

Zmena hesla prebieha odoslaním iba starého a nového hesla metódou **POST** na REST endpoint, ktorý prostredníctvom metód v servisných triedach overuje minimálnu dĺžku nového hesla a zhodu starého hesla s aktuálnym heslom (atribút **password** v triede **User**). V prípade úspechu je heslo zmenené na nové heslo, ktoré je zašifrované. Tento REST endpoint je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/user-controller/changePasswordUsingPOST_1.

2.2.4 Obnova hesla

Obnova hesla je možná prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authentication-controller/changePasswordUsingPOST. V prípade, že zadaný email je platný, vygeneruje sa nové 10-miestne heslo použitím java **UUID.randomUUID()** utility a uloží sa do atribútu **password** triedy **User**. Potom je nové heslo poslané používateľovi emailom z adresy **noreply@story-teller.xyz.**

2.2.5 Obnova registračného emailu

Systém poskytuje taktiež možnosť opätovného zaslania emailu s aktivačným a deaktivačným odkazom. Táto funkcionalita je možná prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authentication-controller/recoverTokenUsingPOST. Táto metóda najprv skontroluje, či existuje používateľ s daným emailom. Potom nasleduje kontrola aktivácie účtu na základe atribútu **activated**. Ak bol účet používateľa s daným emailom aktivovaný, email nie je nutné poslať. Inak sa generuje nový aktivačný token, ktorý sa uloží do atribútu **activationToken**. Nasleduje poslanie emailu na emailovú adresu používateľa tak ako pri registrácii.

2.2.6 Správa prihlásených zariadení

Správa prihlásených zariadení, ktoré sú reprezentované entitou **Token**, bola implementovaná prostredníctvom viacerých REST endpointov. Výber všetkých zariadení, cez ktoré je používateľ prihlásený v aplikácii, je možný prostredníctvom REST endpoint metódou **GET**, ktorý je zdokumentovaný na adrese: https://api.story-teller.xyz/swagger-ui.html#!/user-controller/getAllUserTokensUsingGET. Pre aktuálne prihláseného používateľa sú z databázy vybraté všetky jeho zariadenia.

Pri každom zariadení je možné modifikovať poznámku prostredníctvom REST endpointu metódou **PUT**, ktorý je zdokumentovaný na adrese: https://api.story-teller.xyz/swagger-ui.html#!/token-controller/changeTokenNoteUsingPUT. Poznámka je vyhľadaná na základe atribútu **tokenBody** triedy **Token** a následne je zmenená.

Používateľ môže tiež odstrániť zariadenie zo zoznamu prihlásených zariadení prostredníctvom REST endpointu metódou **DELETE**, ktorý je zdokumentovaný na adrese: https://api.story-teller.xyz/swagger-ui.html#!/token-controller/deleteTokenUsingDELETE. Zariadenie je vyhľadané na základe atribútu **tokenBody** triedy **Token** a atribútu **email** triedy **User** a následne je odstránené.

2.2.7 Riadenie prístupu

Riadenie prístupu k údajom v databáze je realizované pomocou prístupových anotácií **@PreAuthorize.** Tie sú použité pri každom REST endpointe, ktorý pristupuje ku kritickým dátam, špecifickým pre konkrétnych používateľov. Autorizačné anotácie majú tvar: **@PreAuthorize("@roleSercurityService.authorizationMethod(#arg1,'constant1')")**. **@roleSecurityService** reprezentuje triedu, v ktorej sú umiestnené autorizačné metódy. AuthorizationMethod reprezentuje metódu, na základe ktorej je rozhodnuté, či používateľ má prístup k daným údajom. Autorizačné metódy sú definované v triede **RoleSecurityService**, v balíku **com.storyteller.security**. Autorizačná metóda môže prijať argumenty ako **#arg1**, čo je premenná ako napríklad id projektu, id používateľa. Konštanty ako **'constant1'** sú využívané najmä pre roly, keď priradíme REST endpointu konkrétnu rolu, pre ktorú je prístupný.

2.2.8 Načítanie zoznamu používateľov v projekte

Získanie zoradených používateľov prislúchajúcich aktuálne prehliadanému projektu je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese: https://api.story-teller.xyz/swagger-ui.html#!/user-controller/getUsersInProjectUsingGET. Na základe identifikátora triedy **Project** sa vyhľadá projekt a pomocou triedy **ProjectUser** sa získa zoznam používateľov v projekte.

2.2.9 Pridanie používateľskej roly v rámci projektu

Pridanie používateľskej roly v rámci projektu je možná prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese: https://api.story-teller.xyz/swagger-ui.html#!/authority-controller/changeUserAuthoritiesUsingPOST. Na základe identifikátorov tried **User** a **Project** sa vyhľadá objekt typu **ProjectUser** a zmení sa jeho rola. Následne je poslaná notifikácia s nastavenými atribútmi **type, recipient, project** a **body** používateľovi, ktorého rola v projekte bola zmenená.

3 Modul – Lokalizácia

Úlohou modulu lokalizácia je umožnenie nastavenia a vytvorenie prekladu celého systému bez zásahu do funkcionality systému

3.1 Analýza

Na lokalizáciu je možné použiť funkcie jazyka Java a prekladový modul ng-translate pre AngularJS. Samotné preklady je možné ukladať v properties súboroch s názvom v tvare **messages_LOKALIZÁCIA.properties**, ktorý obsahuje preklady v tvare HODNOTA=KĽÚČ.

3.2 Návrh a implementácia

Na implementáciu lokalizácií sme použili prístup popísaný v návrhu. Súbory properties s prekladmi sú umiestnené v **/src/main/resources** a ich obsah si klient preberá cez REST endpoint metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/resource-bundle-controller/listUsingGET. Táto metóda na základe vyžiadaného kódu lokalizácie, napr. sk alebo en, načíta obsah daného properties súboru (v prípade en lokalizácie sa načíta súbor **messages_en.properties**, viď ukážka 1) do objektu typu **Properties**, ktorý je následne serializovaný na JSON a spracovaný modulom ng-translate v klientskej AngularJS aplikácii.

```
myprofile.success.login=You were logged in successfuly
myprofile.password.old=Old password
myprofile.password.new=New password
myprofile.password.new.confirm=Confirm new password
myprofile.update=Update!
myprofile.password.update=Change password!
myprofile.password.old.required=Old password is required
myprofile.password.new.required=New password is required
```

Ukážka 1: Časť súboru messages_en.properties

4 Konfigurácia - Server a nasadzovanie

4.1 Konfigurácia

Náš projekt je založený na architektonickom vzore klient-server, a z toho dôvodu je rozdelený na dva podprojekty. Prvým je projekt **backend**, ktorý sa stará o aplikačnú logiku a perzistenciu. Druhým je projekt **client**, ktorý sa stará o vykresľovanie obsahu a poskytuje používateľovi grafické rozhranie. Oba projekty sú nasadené na servery **Ubuntu Server 16.04**.

Náš server poskytuje tieto domény :

- story-teller.xyz **– tímová stránka o projekte**
- api.story-teller.xyz **– backend aplikácia**
- app.story-teller.xyz **– client aplikácia**
- app.story-teller.xyz **– staging client aplikácia**

4.2 Backend

Projekt backend je založený na **Spring framework-u**, a teda je nasadený na aplikačnom serveri **Apache Tomcat**. Tomcat umožňuje nasadzovanie priamo pomocou URL, to znamená, že priamo po vykonaní buildu je nová verzia nasadená.

4.2.1 Závislosti

Na spúšťanie akceptačných testov sú potrebné nasledujúce nástroje a závislosti:

- Python
  - Python vo verzii > 3.5
  - python3-pip
  - behave
  - selenium
- Linux
  - jdk-8
  - postgresql
  - wkhtmltoimage
  - nodejs
  - npm
  - cordova
  - ionic
  - maven
  - storyteller
  - xvfb
  - tmux
  - ffmpeg
  - firefox
  - geckodriver

- chromium
- chromedriver
- feh
- unclutter

Všetky vyššie uvedené závislosti musia byť umiestnené v ceste PATH. Okrem týchto závislostí je potrebné mať umiestnené skripty **test_definitions** v ceste uvedenej v súbore **pom.xml** a súbor **storyteller-wallpaper.jpg** v ceste **/opt/storyteller-wallpaper.jpg** .

4.3 Client

Client je založený na frameworku **Ionic**, ktorý umožňuje zostaviť aplikácie pre mobilné zariadenia, rovnako ako umiestnenie priamo na webe. Client je nasadzovaný pomocou **SCP** kopírovania a následného spustenia skriptu. Skript sa postará o rozbalenie a nahradenie už existujúcich súborov. Ionic poskytuje službu, ktorá spustí aplikáciu ako webovú stránku, na ktorú sa dá následne pristupovať na základe zvoleného mena.

5 Modul – Projekt

Úlohou modulu projekt má byť možnosť vytvárania nových projektov a spravovania existujúcich projektov prihláseným používateľom.

5.1 Analýza

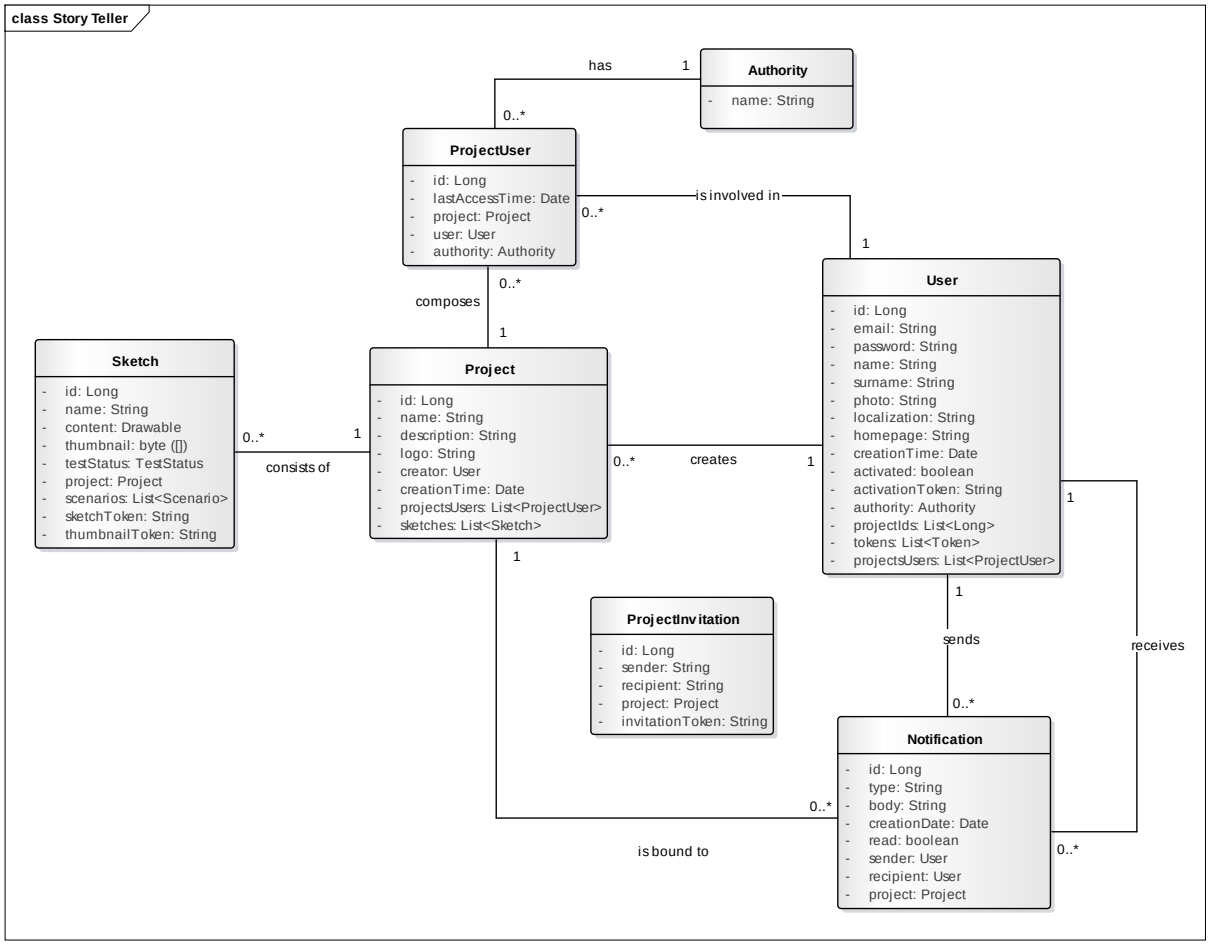
Možnosť vytvárať projekty bude sprístupnená prihláseným používateľom. Pri vytváraní projektu bude používateľ vypĺňať príslušný formulár. Vo formulári bude potrebné zadať názov projektu, opis projektu a prípadne vlastné logo projektu. Následne po odoslaní formulára na spracovanie sa k projektu priradí jeho tvorca – čiže automaticky je tomuto projektu priradený ďalší atribút vyjadrujúci údaje o používateľovi, ktorý projekt vytvoril.

5.2 Návrh a implementácia

Na perzistenciu objektu projektu reprezentovaného triedou **Project** bol použitý OR mapovač s použitím modulu Spring Data.

K základnému prístupu k údajom v databáze boli použité repozitáre rozhrania, ktoré rozširujú rozhrania modulu Spring Data a umožňujú tvorbu tzv. dopytových metód (z angl. query methods), pomocou ktorých modul Spring Data dokáže vygenerovať dopyty do databázy na základe názvu metódy. Repozitáre taktiež podporujú definíciu dopytov pomocou **@Query** anotácií a JPA syntaxe.

Taktiež boli použité servisy (z angl. services) obsahujúce biznis logiku a riadenie databázových transakcií. K dátam sa teda nepristupuje priamo cez repozitáre, ale cez servisy, ktoré následne v transakciách pristupujú k dátam. Týmto je dodržaný princíp viacvrstvovej architektúry.



Obrázok 2: Dátový model projektu

5.2.1 Vytvorenie projektu

Vytvorenie projektu je možné prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/project-controller/createProjectUsingPOST. Nový projekt s nastavenými atribútmi **name, description, logo, creator** a **creationTime** je uložený do databázy. Na validovanie názvu a opisu projektu bola použitá JPA anotácia **@NotNull**.

5.2.2 Načítanie zoznamu projektov prihláseného používateľa

Získanie zoradených projektov prislúchajúcich aktuálne prihlásenému používateľovi je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/project-controller/getUserProjectsUsingGET. Projekty pre aktuálne prihláseného používateľa sú vyhľadané na základe atribútu **email** triedy **User**, pričom je využité stránkovanie.

5.2.3 Načítanie informácií o konkrétnom projekte

Načítanie konkrétneho projektu je realizované načítaním jeho objektu podľa identifikátora. To je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/project-controller/getProjectByIdUsingGET.

Projekt je vyhľadaný na základe identifikátora triedy **Project**. Na strane klienta je tento endpoint volaný už pri zmene obrazovky (stavu) na obrazovku s prehľadom informácií o konkrétnom projekte.

5.2.4 Úprava údajov o projekte

Načítanie konkrétneho projektu je realizované prostredníctvom REST endpointu metódou **PUT**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/project-controller/updateProjectUsingPUT. Projekt je vyhľadaný na základe jeho identifikátora a sú upravené jeho atribúty **name, description** a **logo**.

5.2.5 Priradenie roly používateľovi projektu

Riadenie prístupu ku projektu je založené na rolách. Tie sú používateľovi priradené pomocou REST endpointov, ktoré najprv načítajú všetky dostupné roly, a následne vyznačia aktuálne priradenú. Načítanie všetkých rolí je možné prostredníctvom **REST** endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authority-controller/getAllRolesUsingGET. Načítanie aktuálnej roly používateľa je možné prostredníctvom **REST** endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/authority-controller/getUserProjectAuthorityUsingGET. Najprv sa vyhľadajú objekty triedy **ProjectUser** na základe identifikátorov tried **User** a **Project** a nová rola sa vyhľadá na základe atribútu **name** triedy **Authority**.

5.2.6 Pozvanie používateľa do projektu

6 Používateľské rozhranie

Používateľské rozhranie je realizované v projekte **client**. Ten je založený na technológii **Ionic**, ktorá umožňuje zostavovanie mobilných aplikácií. Framework Ionic ponúka vlastné elementy a štýly na použitie, tie však nie sú príliš vhodné pre webovú stránku. Z toho dôvodu je použitý mobile-first framework **Bootstrap.** V tomto projekte je použitá knižnica **Angular-UI,** nakoľko Ionic využíva AngularJS, ktorý by mohol spôsobovať konflikty s klasickým jQuery.

Bootstrap je responzívny framework, umožňujúci efektívne zobrazenie webovej stránky na rôznych zariadeniach, s rôznym rozlíšením. Dokáže automaticky prispôsobiť rozloženie grafických elemtontov vzhľadom na aktuálnu veľkosť obrazovky. Bootstrap využíva triedy na špecifikáciu štýlu pre príslušný element. V projekte sú využité len štandardné triedy, ktoré využíva bootstrap. Všetky ostatné štýly ako odsadenie, posunutie, a iné, ktoré sú špecifické pre konkrétny element sú definované priamo v HTML. Bootstrap štýly sú definované v súbore **custom.css** a následne minifikované v súbore **custom.min.css.** Iba minifikovaný súbor je importovaný do projektu, a teda pri každej zmene je nutné súbor znovu minifikovať.
Bootstrap dokumentácia : http://getbootstrap.com/getting-started/http://getbootstrap.com/getting-started/

7 Modul – Notifikácie

Úlohou modulu notifikácie je poskytnutie dôležitých informácií používateľovi v reálnom čase, bez nutnosti manuálne vyžiadania.

7.1 Analýza

Na prenos notifikácií je možné použiť websockety s modelom publish – subscribe. Z dôvodu viacerých typov notifikácií je nutné navrhnúť šablónový podmodul, ktorý umožní dynamické nastavenie šablóny konkrétnej notifikácie v závislosti od jej typu, napr. či daná notifikácia bude poskytovať možnosť potvrdenia. Použitie websocketov taktiež umožňuje budúcu implementáciu chatovacieho modulu a modulu paralelného upravovania skíc.

7.2 Návrh a implementácia

Samotná notifikácia je reprezentovaná pomocou triedy **Notification** z balíka **com.storyteller.model**. Databázové operácie nad ňou sú realizované prostredníctvom repozitárov a servisov, podobne ako je to v prípade modulu používateľa. Prístup k notifikáciám z klientskej aplikácie je realizovaný prostredníctvom REST endpointov definovaných v triede **NotificationController** z balíka **com.storyteller.controller**.

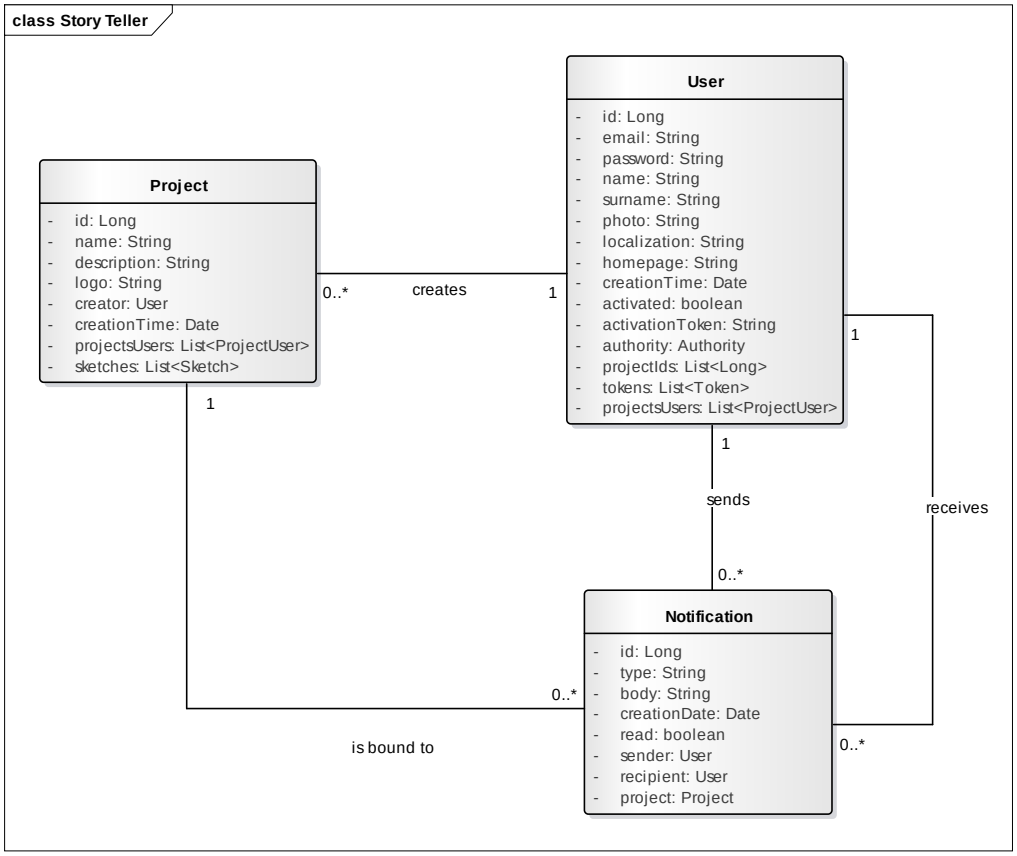
7.2.1 Odosielanie notifikácií

Notifikácie sú odosielané z backendovej strany projektu pomocou protokolu **STOMP**, ktorý je podporovaný rámcom Spring, ktorý používame. Notifikácie sú odosielané prostredníctvom volania metódy **sendNotification(Notification notification)**, do ktorej vstupuje už vyplnený objekt notifikácie vrátane príjemcu. Táto metóda volá Spring metódu **convertAndSendToUser(String user, String destination, Object payload)**. Argument **user** predstavuje identifikátor používateľa, v našom prípade jeho email. Argument **destination** predstavuje cestu websocketového publish kanálu, do ktorého sa má notifikácia zverejniť. Táto cesta musí obsahovať prefix **'/topic'** z dôvodu odlíšenia publish ciest od subscribe. Argument **payload** predstavuje samotný obsah správy, t. j. notifikáciu. Táto metóda je volaná prostredníctvom objektu **SimpMessageSendingOperations**, ktorý je súčasťou Spring modulu pre websockety. Autentifikácia používateľa pre websockety je realizovaná pomocou objektu **AuthenticationPrincipal**, ktorý je vytvorený pred samotným odoslaním správy do konkrétneho kanála, t. j. po prijatí dispatcherom.

7.2.2 Prijímanie notifikácií

Notifikácie sú prijímané s využitím knižnice **SockJS** a javascriptového modulu pre protokol **STOMP**. Knižnicu **SockJS** bolo potrebné upraviť z dôvodu **CORS** mechanizmu – knižnica neposkytuje možnosť vypnúť atribút **withCredentials** pre **XMLHttpRequest** volania a rovnako sa neriadi globálnymi nastaveniami rámca AngularJS. Bolo preto nutné nastaviť atribút **withCredentials** na **false**. Prijímanie notifikácií zabezpečuje subscribe na kanál **'/user/topic/notifications'** v metóde **onConnect(frame)** komponentu **socketService**. Tento

komponent predstavuje wrapper nad knižnicou **SockJS**, ktorý bolo nutné vytvoriť z dôvodu chýbajúcej podpory autentifikácie prostredníctvom tokenu.



Obrázok 3: Dátový model notifikácií

Po prijatí správy je notifikácia pretransformovaná z JSON reťazca na objekt, následne je zobrazená a je broadcastnutý event '**$newNotification**', čo umožňuje spracovanie notifikácie aj ďalším modulom, ako je napríklad navigačný modul. Tento modul inkrementuje počítadlo zatiaľ nevidených notifikácií v navigačnom panely nachádzajúcom sa na vrchu aplikácie a indikuje novú notifikáciu animáciou zatrasenia ikony notifikácií.

7.2.3 Zobrazenie notifikácií

Notifikácie sú zobrazované ako **toast popup** prostredníctvom modulu **ngToaster**. Na zobrazenie sa používa metóda **showNotification(notification)** komponentu **notificationService**, do ktorej ako argument prichádza vyplnená notifikácia určená na zobrazenie.

Taktiež je možné prezerať notifikácie cez otváracie popup okno, ktoré sa zobrazí kliknutím na ikonu notifikácií v navigačnom panely – volanie metódy **showPreviewNotifications()** z komponentu **navigationController**. Táto ikona taktiež zobrazuje počet neprečítaných notifikácií, čo je realizované volaním metódy **getUnreadNotificationsCount()** z komponentu **navigationController**. Po kliknutí na ikonu sa zobrazí päť najnovších notifikácií, ktoré sú načítané volaním metódy **getBriefNotifications()** z komponentu **navigationController**.

K starším notifikáciám je možné pristupovať cez tlačidlá pre stránkovanie. Stránkovanie je realizované volaním metódy **getNotificationPage(page)** z komponentu **navigationController** do ktorej ako argument vstupuje číslo strany, ktorá sa má zobraziť.

Označovanie notifikácií za videné prebieha tak, že notifikácie, ktoré boli zobrazené cez otváracie popup okno sú odoslané backendovej aplikácií, ktorá im zmení príznak **read** na true. Toto je realizované metódou **setNotificationsAreRead()** z komponentu **navigationController**. Po úspešnom odoslaní je tento príznak nastavený aj daným notifikáciám v klientskej časti a počítadlo neprečítaných notifikácií je aktualizované.

8 Modul – Skice

Možnosť vytvárať skice je jednou z najdôležitejších funkcionalít v tomto projekte. Pomocou skíc je možné prototypovať používateľské rozhranie, na základe ktorého budú neskôr generované akceptačné testy.

Používateľské rozhranie na webových stránkach býva zložené z viacerých komponentov, ktoré slúžia ako ovládacie prvky, alebo ako tzv. kontajnery, do ktorých sa potom vkladajú iné prvky.

Dôležitým aspektom pri skladaní rozhrania pomocou komponentov je určovanie pozícií jednotlivých prvkov. Prvky môžu mať absolútnu pozíciu, kde ich pozícia je počítaná od kraja plátna, na ktorom sú umiestnené, alebo môžu mať relatívne pozície, pri ktorých sa pozície počítajú vzhľadom na 'rodiča'.

8.1 Analýza

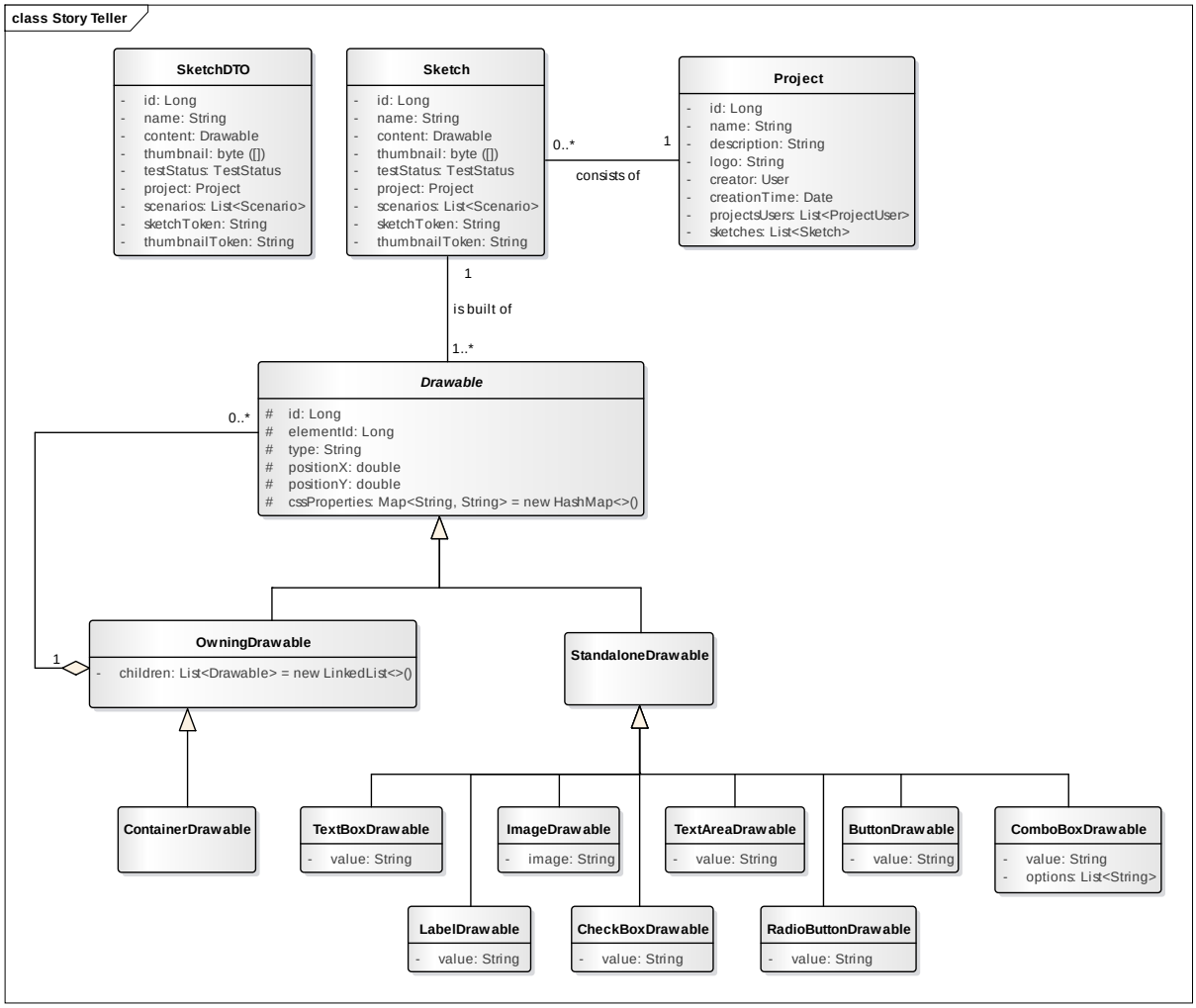
Pre možnosť tvorby skice je nutné vytvoriť používateľské rozhranie, ktoré bude poskytovať plátno a panel nástrojov, z ktorého sa budú môcť pridávať prvky na plátno.

Prvky používateľského rozhrania majú vo všeobecnosti veľkú časť vlastností spoločných. Ide napríklad o ich pozíciu na plátne, farbu, veľkosť, a mnoho iných. Najefektívnejším spôsobom je vytvoriť všeobecnú abstraktnú triedu, ktorá bude tieto vlastnosti obsahovať, a konkrétne prvky používateľského rozhrania budú od tejto abstraktnej triedy dediť, pričom ich vlastné atribúty budú len tie, ktoré sú špecifické pre konkrétny prvok.

8.2 Návrh a implementácia

8.2.1 Perzistencia

Dátový model skice a jej komponentov je zobrazený na obrázku 4. Trieda **Sketch** predstavuje samotnú skicu, pričom atribút **thumbnail** predstavuje generovaný náhľad a atribút **content** jej obsah, t. j. zloženie z komponentov. Komponenty skice sú navrhnuté ako vzor composite, pričom komponent predstavuje abstraktná trieda **Drawable**. Od tejto triedy dedia triedy **OwningDrawable**, ktorá predstavuje komponent, ktorý môže mať ďalšie vnorené prvky, čo v kontexte vzoru composite predstavuje časť composite. Príkladom komponentu, ktorý dedí od tejto triedy, je komponent kontajner. Ďalšou dediacou triedou je trieda **StandaloneDrawable**, ktorá predstavuje samostatný komponent, t. j. komponent, ktorý nemôže obsahovať ďalšie vnorené komponenty. Príkladom je komponent textové pole, ktorý je reprezentovaný triedou **TextBoxDrawable** so špecifickým atribútom **value**, čiže jeho hodnotou. Všetky tieto triedy obsahujú metódu na vygenerovanie inline CSS s názvom **getInlineCss()**, ktorá vracia inline CSS ako reťazec na základe obsahu poľa **cssProperties**. Taktiež obsahujú metódu **toHtml()**, ktorá vracia html reprezentáciu jednotlivého prvku ako reťazec, čo znamená, že v prípade zavolania tejto metódy na inštanciu triedy **Drawable** s typom (pole type) **ROOT_CONTAINER** sa vytvorí html reprezentácia celej skice. Takéto volanie je použité napríklad pri generovaní náhľadu skice. Trieda **SketchDTO** predstavuje pomocnú triedu, ktorá má rovnaká atribúty ako trieda **Sketch**.



Obrázok 4: Dátový model skice a komponentov

8.2.2 Vytvorenie skice

Vytvorenie skice je možné prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/saveSketchUsingPOST. Skica a jej komponenty sú uložené do atribútov **name, content, thumbnail, scenarios, testStatus** triedy **Sketch** a projektu, ktorý je vyhľadaný na základe jeho identifikátora, je priradená vytvorená skica. Súčasťou vytvorenia skice je automatické generovanie náhľadu vo forme obrázku. Na to je použitá knižnica **wkhtmltopdf**, resp. jej časť **wkhtmltoimage**. Keďže táto knižnica nie je priamo kompatibilná s Javou, bol vytvorený (upravený) wrapper, ktorý túto knižnicu volá prostredníctvom vytvorenia nového procesu a dočasného html súboru so skicou, ktorý je špecifikovaný ako prvý argument volania príkazu **wkhtmltoimage**. Výstup (vyrenderovaný obrázok) je presmerovaný cez deskriptor do bajtového poľa v Jave.

8.2.3 Získanie skice

Získanie skice podľa jej identifikátora je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-

ui.html#!/sketch-controller/getSketchUsingGET. Táto metóda vyberie skicu z databázy na základe identifikátora aktuálneho projektu.

8.2.4 Úprava skice

Úprava skice prebieha odoslaním celého objektu skice a identifikátora skice do REST endpointu metódou **PUT**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/updateSketchUsingPUT. Získa sa skica na základe jej identifikátora a nové informácie o skici sú uložené do databázy, ide o atribúty **name, content, thumbnail, scenarios, testStatus**.

8.2.5 Odstránenie skice

Systém poskytuje tiež možnosť odstránenia skice, čo bolo implementované prostredníctvom REST endpointu metódou **DELETE**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/deleteSketchUsingDELETE. Najprv sa získa skica na základe jej identifikátora a nájdená skica je odstránená z databázy.

8.2.6 Načítanie skíc projektu

Načítanie skíc projektu bolo implementované prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/getSketchesByProjectIdUsingGET. Získa sa zoznam skíc na základe identifikátora projektu.

8.2.7 Export layoutu skice

Export layoutu skice bolo implementované prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/getSketchLayoutForTypeUsingPOST. Na základe vstupného parametra, ktorý označuje typ layoutu, sa vygeneruje HTML alebo Android layout pre skicu vyhľadanú na základe jej identifikátora. Vygenerovaný layout je automaticky stiahnutý pre klienta.

8.2.8 Načítanie náhľadu skice

Načítanie náhľadu skice je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/getSketchThumbnailUsingGET. Na základe identifikátora je vyhľadaná skica a vráti sa objekt typu **SketchDTO** s nastavenými atribútmi **id** a **thumbnail**.

8.2.9 Zdieľanie skice

Zdieľanie skice je možné pomocou vygenerovaných odkazov na zdieľaný náhľad skice alebo zdieľanú skicu. Táto funkcionalita bola implementovaná prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/generateRouteUsingPOST. Pre zdieľanie skice sa získa skica na základe jej identifikátora a je vrátená objekt typu **SketchDTO** s nastavenými vygenerovanými

atribútmi **sketchToken** alebo **thumbnailToken**, podľa toho, či ide o zdieľanú skicu alebo zdieľaný náhľad skice.

8.2.10 Načítanie zdieľaného náhľadu skice

Systém poskytuje tiež možnosť načítania zdieľaného náhľadu skice, čo bolo implementované prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/getSharedThumbnailUsingGET. Skica je načítaná na základe atribútu **thumbnailToken** a vráti sa objekt typu **SketchDTO** s nastavenými atribútmi **name** a **thumbnail**.

8.2.11 Načítanie zdieľanej skice

Systém poskytuje tiež možnosť načítania zdieľanej skice, čo bolo implementované prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/getSharedSketchUsingGET. Skica je načítaná na základe atribútu **thumbnailToken** a vráti sa objekt typu **SketchDTO** s nastavenými atribútmi **name, content, scenarios, thumbnailToken** a **sketchToken**.

9 Modul – Scenáre a testy

Na vykonávanie akceptačných testov je nutné mať k dispozícii okrem skice aj scenár testov. Scenáre testov pozostávajú z individuálnych krokov, ktoré na seba často nadväzujú a v prípade akceptačných testov týkajúcich sa softvérových systémov tieto kroky predstavujú akcie používateľa. Úlohou scenárov je teda obohatiť vytvorenú skicu o akcie a transformovať statickú skicu do vykonateľnej podoby.

9.1 Analýza

Scenáre pozostávajú z jednotlivých krokov – akcií. Tieto akcie sú naviazané na vizuálne elementy skice a teda mali by byť pridávané súčasne s vizuálnymi prvkami. Vykonávanie scenárov by malo používateľovi poskytovať informáciu o stave vykonávania a upozorniť ho v prípade, že scenár nemohol byť úspešne vykonaný. Toto je možné zabezpečiť buď jednoduchým stavovým indikátorom, alebo videozáznamom z testovania.

9.2 Návrh a implementácia

9.2.1 Perzistencia

Dátový model skice a jej komponentov je zobrazený na obrázku 05. Trieda **Scenario** predstavuje samotný scenár, pričom atribút **gherkinFeatureContent** predstavuje vyjadrenie scenáru v notácii Gherkin. Tento atribút je tranzitívny a teda nie je perzistovaný do databázy – je využitý len na prenos tejto informácie na stranu klienta. Scenár pozostáva z krokov, ktoré sú reprezentované triedou **ScenarioStep**. Táto trieda pozostáva z vyjadrenia kroku a typu kroku – akcie vhodnej pre notáciu Gherkin. Taktiež obsahuje informáciu o poradí v scenári. Trieda **StepState** sa vzťahuje na triedu **ScenarioStep** a vyjadruje stav pre krok scenára. Trieda **TestResult** predstavuje informáciu o vykonávaní akceptačných testov. Okrem iného obsahuje stav testu, reprezentovaný pomocou vymenovaného typu **TestStatus**, dĺžku trvania a cestu k súboru s videozáznamom testu. Taktiež obsahuje informáciu o webovom prehliadači a rozlíšení, v ktorom bol daný test vykonávaný. Dostupné webové prehliadače sú reprezentované triedou **Browser** a dostupné rozlíšenia sú reprezentované triedou **Resolution**.

9.2.2 Vytvorenie, odstránenie a získanie scenáru

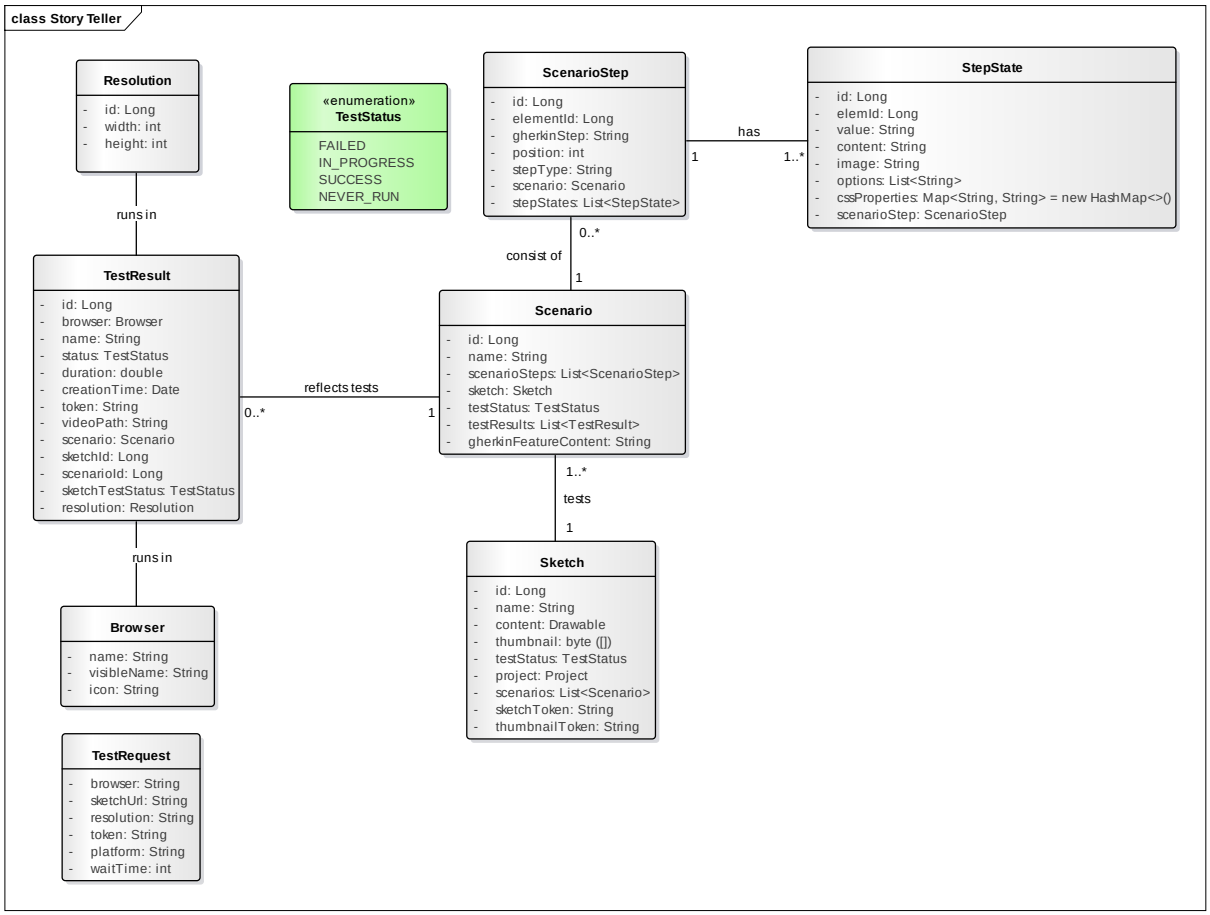
Vytvorenie scenáru, a teda jeho krokov, prebieha súčasne s vytváraním skice. Kroky sú pridávané v závislosti od pridaného vizuálneho prvku skice. Každému prvku je možné nastaviť meno a definovať obmedzenie na jeho vlastnosť. Obmedzenie sa vždy začína znakom „<“ a končí znakom „>“ a musí byť v presnom formáte. Implementované sú tieto obmedzenia:

- <not-empty> - vlastnosť prvku nesmie byť prázdna,
- <length>N> - dĺžka vlastnosti prvku musí byť väčšia ako N, kde N je kladné číslo, namiesto znaku „>“ môžu byť použité znaky „<“ a „=“,
- <email> - vlastnosť prvku musí mať tvar emailu.

Odstránenie krokov prebieha taktiež súčasne s odstránením vizuálnych prvkov skice – ich presunutím z plátna na ikonu zobrazujúcu kôš. Následná perzistencia scenáru je

zabezpečená spolu s perzistenciou skice a teda prostredníctvom REST endpointu metódou **POST**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/saveSketchUsingPOST, a pri aktualizácii prostredníctvom REST endpointu metódou **PUT**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/sketch-controller/updateSketchUsingPUT. Ich funkcionalita je popísaná v časti 8.2.

Získanie scenáru je možné pri získaní skice a prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/scenario-controller/getScenarioByIdUsingGET. Na základe identifikátora scenára je scenár vyhľadaný a vrátený.



Obrázok 5: Dátový model scenárov a testov

Získanie Gherkin notácie scenáru je možné prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/scenario-controller/getGherkinFeatureByScenarioIdUsingGET. Skica je určená svojím identifikátorom a sú vygenerované a vrátené Gherkin notácie.

9.2.3 Vykonávanie testov

Spustenie testu pre konkrétny scenár je možné prostredníctvom REST endpointu metódou **POST**, pričom takýto test je spustený v prednastavenom webovom prehliadači PhantomJS a prebieha na webovej adrese zadanej pomocou parametra **url**. Tento endpoint je

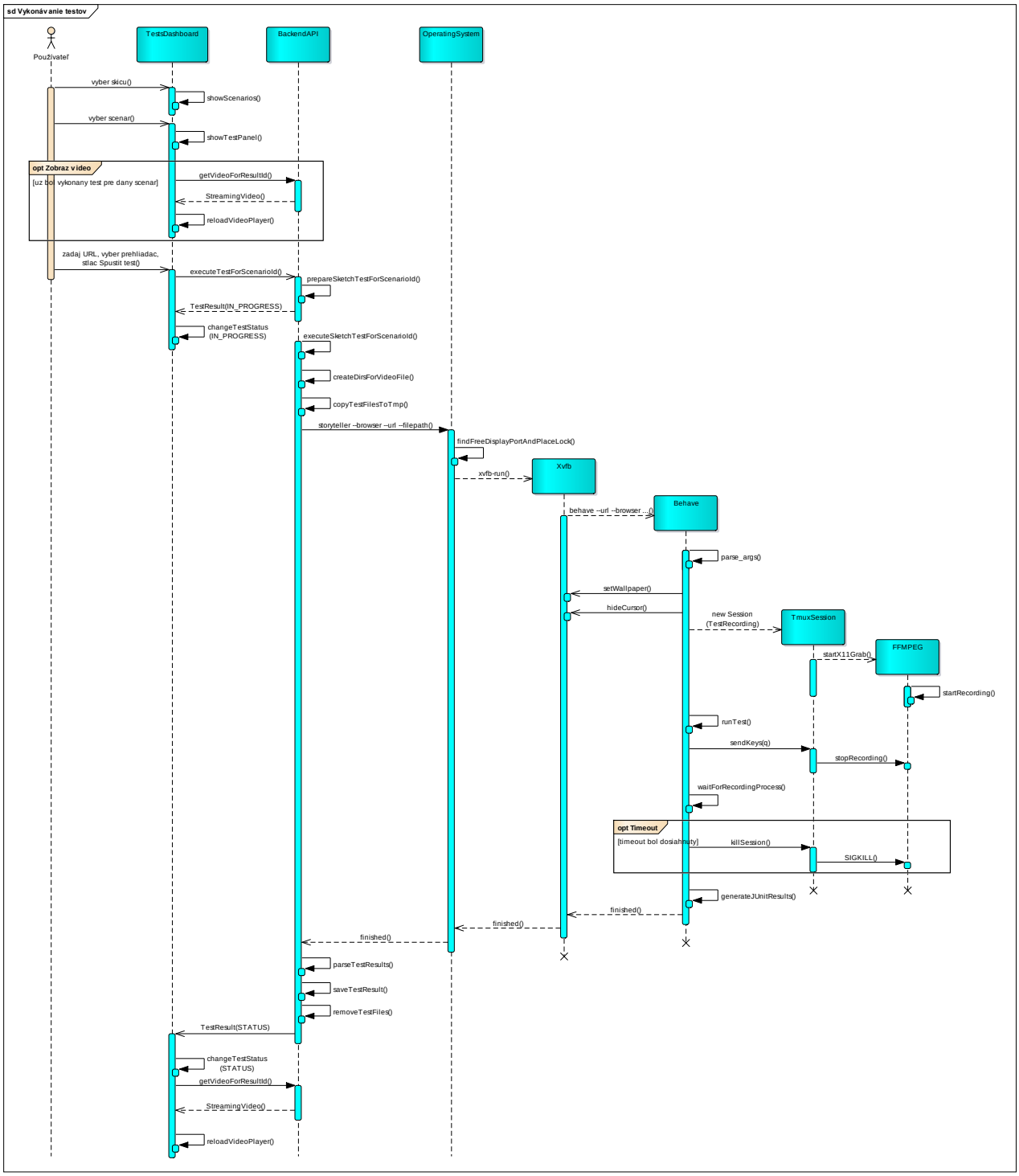
zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/scenario-controller/executeTestForScenarioIdWithDefaultBrowserUsingPOST.

Spustenie testu v inom ako prednastavenom prehliadači je možné prostredníctvom REST endpointu metódou **POST**, pričom parametre požiadavky – **browser**, **url** a **resolution** špecifikujú použitý webový prehliadač, webovú adresu a rozlíšenie. Tento endpoint je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/scenario-controller/executeTestForScenarioIdWithDefaultBrowserUsingPOST. Dostupné webové prehliadače sú načítané prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/test-controller/getAvailableBrowsersUsingGET. Dostupné rozlíšenia sú načítané prostredníctvom REST endpointu metódou **GET**, ktorý je zdokumentovaný na adrese https://api.story-teller.xyz/swagger-ui.html#!/test-controller/getAllResolutionsUsingGET.

Priebeh vykonávania testov je opísaný na obrázku 6. Pred samotným spustením testu sa predpokladá, že používateľ je prihlásený, má vybratý projekt a nachádza sa v časti **akceptačné testy**.

Po vybratí webového prehliadača, zvolení webovej adresy stránky na testovanie a stlačení tlačidla na spustenie testu sa na stranu servera odošle požiadavka na začatie testu. Následne sa vytvorí inštancia objektu **TestRequest**, ktorá sa odošle na jeden z uzlov zodpovedných za vykonávanie modulárnych testov. Tento uzol na základe tokenu stiahne testovacie súbory, ktoré mu pripraví server v zip archíve a spustí test v docker kontajneri. Na strane servera pripraví nová inštancia objektu **TestResult**, vyplnia sa jej polia, uloží sa do databázy a nastaví sa stav daného testu na **IN PROGRESS**. Samotný test prebieha v izolovanom docker kontajneri, ktorý zavolá skript **storyteller** s parametrami a otestuje stiahnuté testovacie súbory. Tento skript následne vytvorí inštanciu virtuálnej **Xvbf** obrazovky, v ktorej spustí testovací nástroj **behave**. Ten následne spustí nahrávanie testu pomocou nástroja **ffmpeg** v samostatnej **tmux** relácií. Po vykonaní testu **behave** odošle relácii **tmux** pokyn na zastavenie nahrávania videa. Pokiaľ toto nahrávanie neskončí do určitého času po odoslaní pokynu na zastavenie, nástroj **behave** odošle pokyn relácii **tmux** na vynútenie zastavenia nahrávania – relácia **tmux** vtedy odošle nástroju **ffmpeg** signál **KILL**.

Po ukončení nahrávania nástroj **behave** vygeneruje testovú správu v podobe JUnit správy, ktorý odošle hlavný uzol spolu s videozáznamom testu na server, ktorý tieto súbory spracuje a uloží v podobe aktualizovaného objektu **TestResult**. Tento objekt následne odošle na klientsku časť. Klientska časť tento objekt spracuje, aktualizuje stav vykonávaného testu a načíta videozáznam z testovania.



Obrázok 6: Priebeh testu