ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

KIV/JET

Konverze RAMSES Akademie
na single page aplikaci s REST backendem

Autoři: Antonín NEUMANN, A14N0139P
Martin BLÁHA, A14N0119P
David KOŠEK, A14N0132P

Akademický rok: 2014/2015

1 Kontext

Cíle projektu

- Vytvořit jednoduchou aplikaci typu single page s RESTovým backendem.
- K aplikaci budou mít přístup pouze autentikovaní uživatelé.
- Backend aplikace musí být dokumentován podle standardů pro RESTové služby

2 Technologie

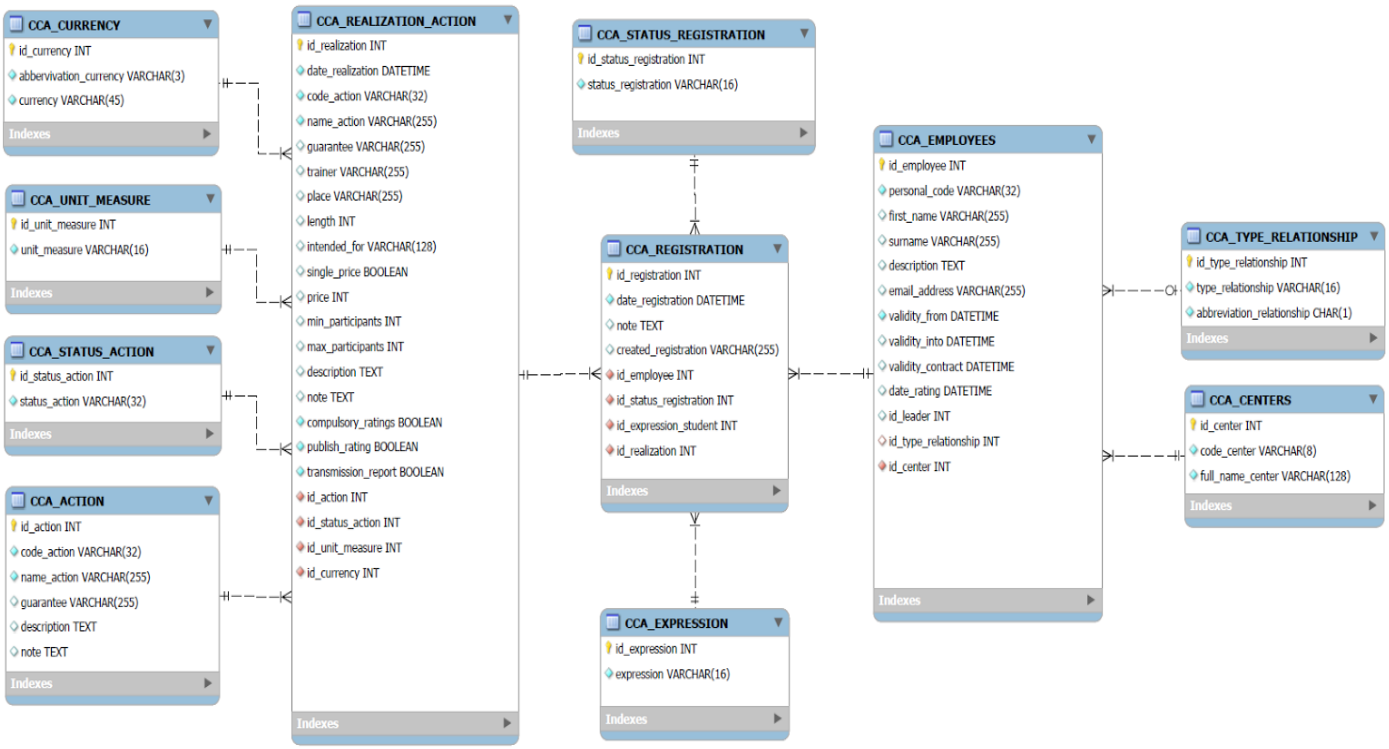
Technologie pro vytvoření aplikace byly zadány:

- Java EE 7 (JPA 2.1, JAX-RS 2.0, EJB 3.2)
- Aplikační server GlassFish 4.1
- ORM - JPA
- REST API - Jersey
- Maven 3
- Databáze MySQL 5.6+
- Javascript AngularJS 1.3
- Git

3 Architektura

3.1 Databázový model

Databázový model (ERA) je zobrazen na obrázku (obr. 3.1). Jednotlivé entity mají jednoznačný název, který naznačuje, čeho se daná entity týká. Všechny položky v entitě byly voleny dle logického uvážení s případnou konzultací vedoucího práce. Model má 11 entit včetně jednoduchých číselníků.



Obrázek 3.1: Databázový model.

3.1.1 Popis vazeb

V modelu se nachází především vazby 1:N. Vazby jsou zřejmé a jasné (například vazba mezi entitami CCA_EMPLOYEES a CCA_CENTER je použita zmíněná vazba 1:N a říká nám, že v jednom středisku může pracovat více zaměstnanců a jeden zaměstnanec může pracovat pouze v jednom centru). Popíšeme si pouze jedinou vazbu M:N mezi entitami CCA_EMPLOYEES a CCA_REALIZATION_ACTION nám vznikla rozkladová tabulka CCA_REGISTRATION. Zaměstnanci (externí i interní) podávají přihlášky na vypsané realizace různých akcí a je zřejmé, že více osob se může přihlásit na tutéž realizaci.

4 Technické řešení

Implementaci aplikace lze rozdělit na dvě větší části. První část je prezenční vrstva (AngularJS) a druhá část je business vrstva (JavaEE).

4.1 Business vrstva

Tato vrstva se zabývá především komunikací mezi databází a prezentační vrstvou na základě požadavků. Při vývoji této aplikace jsme použili hodně anotací jak pro RESTové služby, tak pro mapování databáze (JPA). Jak již bylo řečeno, z prezenční vrstvy přijde požadavek, který je zpracován, existuje-li k němu daná cesta a následně je zpět odeslána odpověď s požadujícími daty ve formátu JSON.

4.1.1 Objektově relační mapování

Objektově relační mapování vychází z myšlenky vytvoření jednoho řádku databázové tabulky jako objektu v aplikaci pro snazší práci s daty. K tomu využíváme JPA a pomocí anotací jsme vytvořili jednotlivé entity jako jednotlivé třídy. U každé této třídy jsou namapovány všechny sloupečky dané entity, ke kterým jsou vygenerovány tzv. settery a gettery pro přístup, resp. nastavení jednotlivých hodnot. Je-li k entitě navázána některá z vazeb (1:N, M:N), tak je také namapována a je vytvořen seznam daných objektů. Každá z těchto tříd musí mít bez-parametrový konstruktor pro vytvoření nového řádku (objektu). U každé třídy jsou uchovány i jednotlivé databázové dotazy pro získání dat, které jsou volány z jednotlivých implementacích DAOBean.

4.1.2 DAO a DAOBean

DAO jsou obecně označovány třídy pro přístup do databáze. Z jiného místa, než prostřednictvím DAO instancí by se nemělo do databáze přistupovat. DAO jsou poté označovány jako interface a DAOBean jako již konkrétní implementace daného interface. Všechny implementace DAO jsou anotovány anotací @Stateless, což nám říká, že třída představuje bezstavový bean. Pro obsloužení každého požadavku klienta je mu vždy na serveru přidělena samostatná instance, která je vyhrazena pouze tomuto klientovi v rámci jednoho daného požadavku – z tohoto důvodu jsou bezstavové beany přirozeně vláknově bezpečné. V každé DAOBean si vytvoříme EntityManagera anotovaného anotací @PersistanceContext. Pomocí tohoto EntityManagera jsme schopni provolávat jednotlivé požadované dotazy, jako jsou uložení nového záznamu, smazání záznam, update stávajícího záznamu atd.

4.1.3 Autentikace

Pro využití RESTových služeb je nejdříve provést autentikaci. Autentikace se provádí posláním HTTP požadavku metodou POST na adresu /login, kde v proměnné Authorization je uživatelské jméno a heslo oddělené dvojtečkou zakódované do Base64 (jedná se o standardní HTTP Base Authorization - RFC 2617). Na serveru je tento požadavek předán aplikačnímu serveru, který uživatele ověří pomocí file realmu. Pokud uživatel existuje, server vygeneruje pomocí UUID 32 znakový řetězec, který bude sloužit jako token pro klienta. Tento token je na serveru uložený s údajem, jestli jde o uživatele s právy user nebo admin. Token a práva jsou následně odeslány klientovy v HTTP požadavku jako proměnné X-Token a X-Role. Při každém požadavku, který vede na adresu /api/* je potřeba, aby klient měl v HTTP požadavku uveden token v hlavičce X-Token, v opačném případě server klientovy odpoví, že uživatel je neautentizovaný.

4.1.4 REST

RESTové služby přestavují obsluhu požadavků GET, POST, PUT a DELETE. Pro volání těchto požadavků pro jednotlivé “entity” musí být vytvořeny samostatné třídy. Která z těchto tříd se má provolat vychází vždy z požadované URI z PL (Presentation Layer), proto má každá z těchto tříd anotaci @Path (například RESTové služby požadované nad akcemi mají anotaci @Path(“action”)). Poté následují jednotlivé metody označeny opět pomocí anotací @GET, @POST atd.. Některé můžou mít i zmíněnou anotaci @Path, která nám říká, pod jakou URI se metoda zavolá. V případě, že má metoda vstupní parametry jsou označovány anotací @PathParam a v dané URI jsou ve složených závorkách (například pro zobrazení akce je @Path("/detail/{id}") a v metodě je uveden parametr jako @PathParam("id") int id). Dále jsou použity anotace @Produces a @Consumes, který nám udávají, co která metoda zpracovává a co vrací.

4.2 Prezentační vrstva

Prezentační vrstva je napsaná v Javascriptovém frameworku AngularJS 1.3. Veškeré moduly, kontrolery a šablony jsou spolu v jedné podložce - jedná se tedy o tzv. rozdělení “by feature”.

Každý modul obsahuje své vlastní routování URL adres. V jednotlivých kontrolerech dochází k volání REST API pro získání aktuálně potřebných dat. Na straně klienta se ukládá pouze token, který je potřebný pro komunikaci s API - jinak se jedná o tzv. stale less frontend.

5 Závěr

Projekt pro externí firmu CCA Group a.s. se zabýval převodem klasické serverové aplikace pro správu vzdělávání zaměstnanců. Tato původní aplikace má stateful backend a uživatelské rozhraní generované na serveru při každém požadavku od uživatele.

Požadovaným cílem byl návrh a implementace single page aplikace s REST backendem, která by stávající aplikaci mohla nahradit a zároveň by sloužila jako proof of concept pro zařazení těchto nových technologií a principů do standardního vývojového stacku firmy pro nadcházející projekty.

Aplikace splňuje zadání a je navržena a implementována pomocí požadovaných technologií. Při práci jsme se snažili dodržovat domluvenou jednotnou konvenci včetně psaní kódu v anglickém jazyce (včetně komentářů).

Při práci jsme se seznámili s většinou nových technologií, které nám byli doposud implementačně neznámé. Prohloubili jsme si znalosti s datovým úložištěm GIT. Pro aplikaci jsme zvolili databázový systém MySQL z důvodu, aby každý vývojář mohl pracovat ve svém lokálním prostředí bez závislosti na internetu.

Týmové role jsme si určili na začátku a byly dodržovány po celou dobu vývoje aplikace.