Semestrální práce z předmětu KIV/SAR

Centrální evidence osob

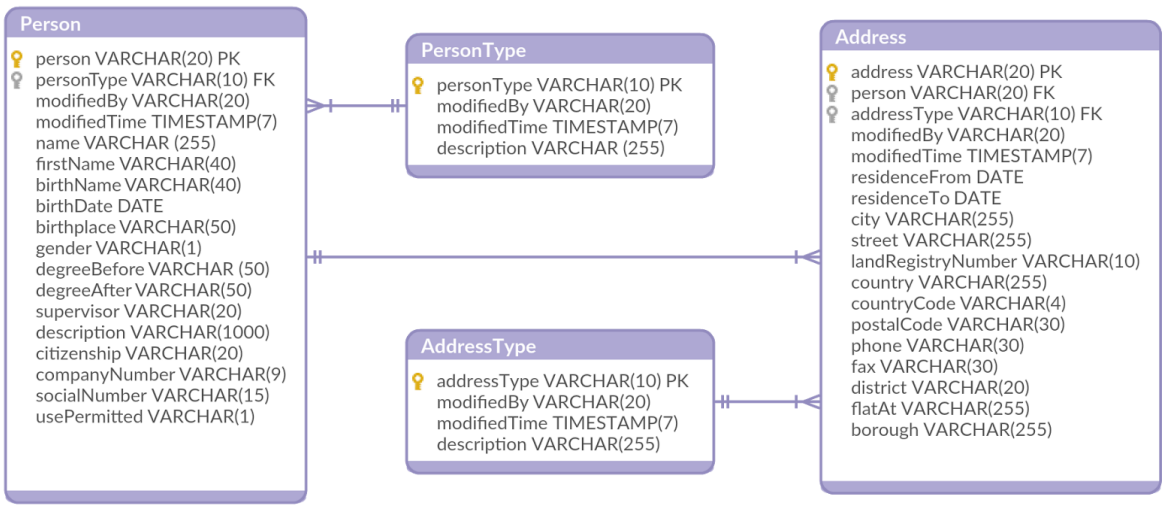
Petr Dobřička (dobripet@students.zcu.cz)
Matěj Lochman (lochmanm@students.zcu.cz)
Marek Rasocha (rasocham@students.zcu.cz)
V Plzni 4.1.2017

Stávající stav

- Seznam je na 200 organizacích, každá má svoji vlastní databázi a aplikace na ní přistupující.
- Každá organizace spravuje osoby nezávisle na ostatních (duplicitní osoby přes všechny databáze)
- Problém – nutno zachovat stávající identifikátory (klidně vedle nových, pokud bude třeba)
- Rozsah 1milion záznamů osob a jejich adres.

Původní architektura

Protože je několik aplikací(i třetích stran), které přistupují přímo do databáze a nelze je momentálně měnit, tak jediné místo, kde může být provedena úprava je databáze. Stávající architektura je tudíž pouze databáze s daty, viz Obrázek 1. Velice se hodí, že všechna data již mají příznak kdo a kdy je naposledy upravil.



Obrázek 1: Původní schéma databáze.

Požadavky

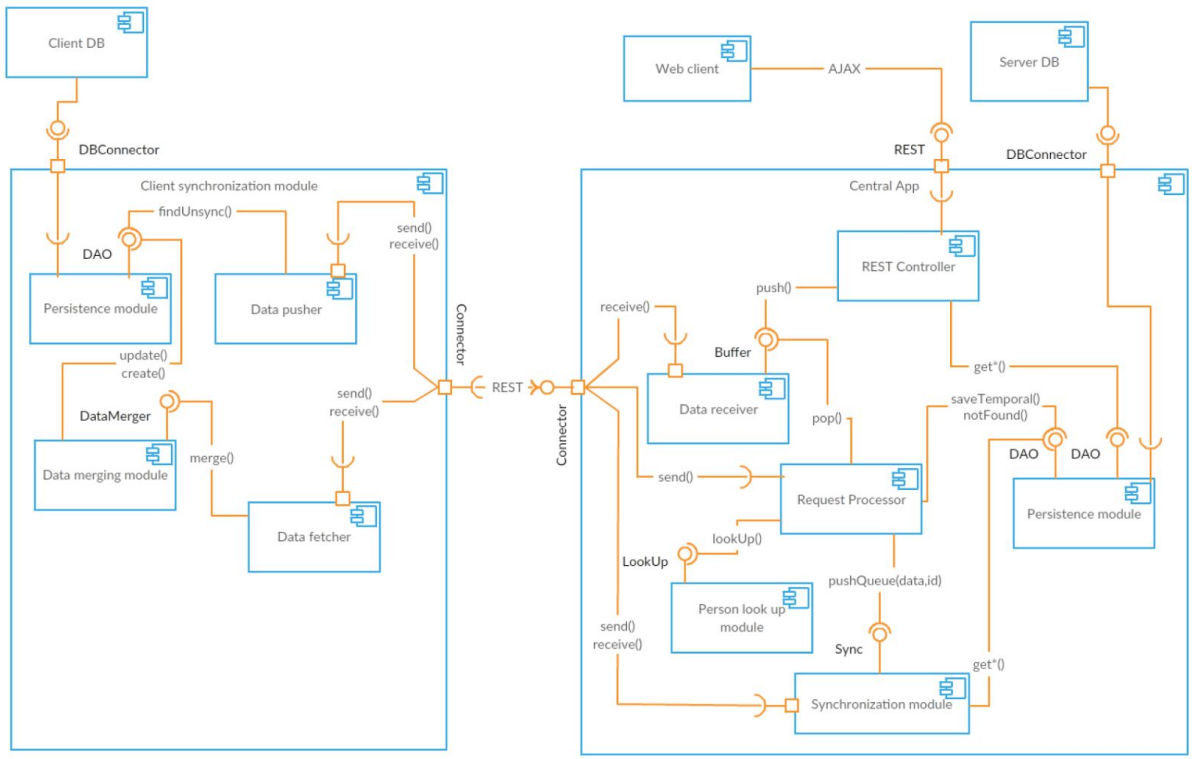
- Zákazník touží po jednom seznamu a eliminaci duplicit, tudíž jedna centrální databáze, která v sobě bude uchovávat neduplicitní osoby ze všech organizací.
- Technologickou nezávislost rozhraní webových služeb.
- Uživatelské rozhraní pro změny a prohlížení centrální evidence.
- Uživatelé budou s přístupem zvlášť na zápis a zvlášť na prohlížení.
- Webové služby zatím jednoduchá autentikace proti vyjmenovaným uživatelským účtům.
- Osoby budou nejprve dočasné, po simulované kontrole proti ISZR budou skutečné.
- Osoby budou uložené v centru i na lokálech, budou synchronizované službami.

Vybrané technologie

- Spring
- PostgreSQL
- AngularJS
- REST

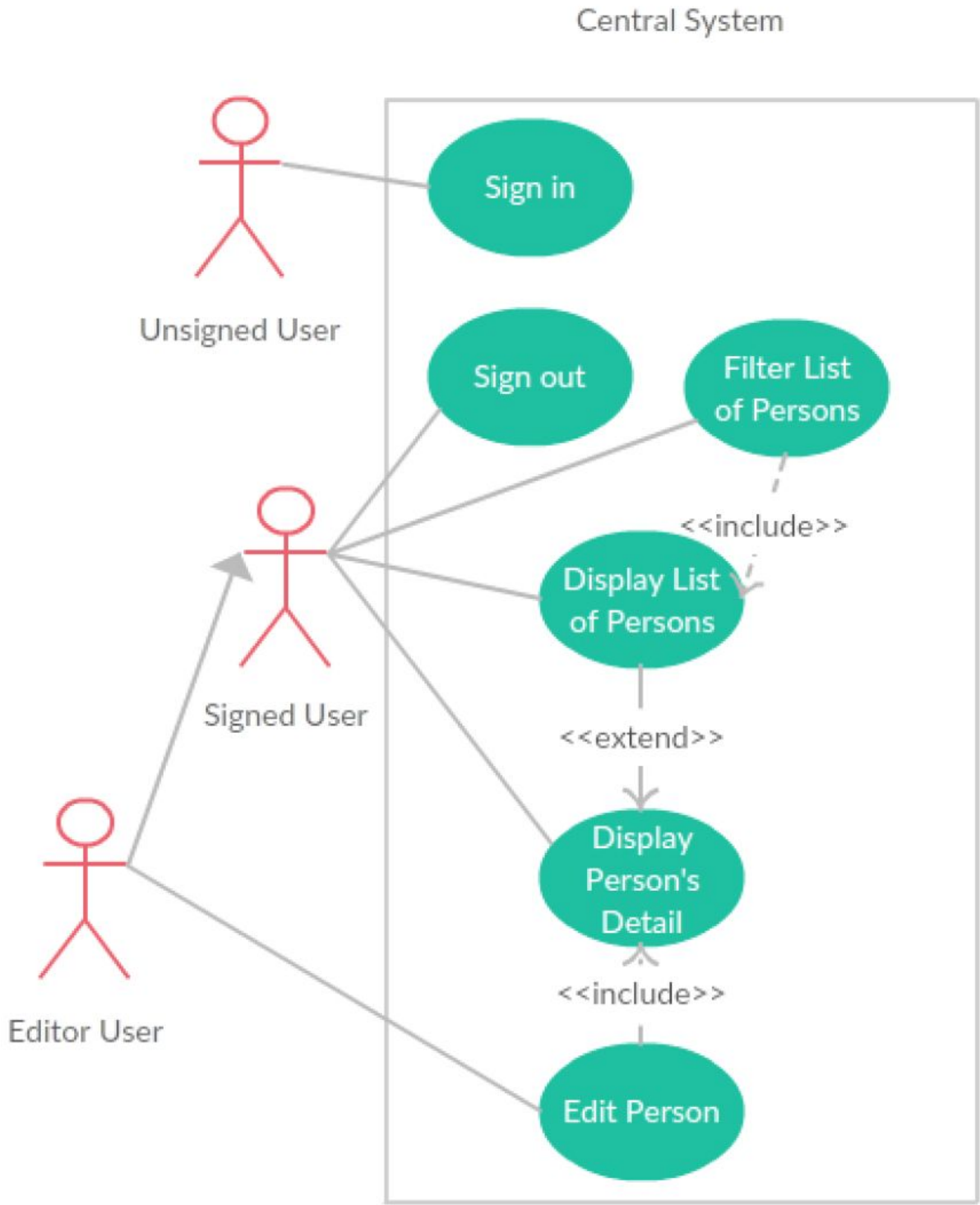
Návrh řešení

Pro tento systém jsme navrhli následující řešení. Na klientské a serverové části bude několik modulů, které spolu budou komunikovat přes rozhraní. Tento pohled je zachycen na Obrázku 2. Na institucích poběží klientské moduly a v centrálním serveru serverové.



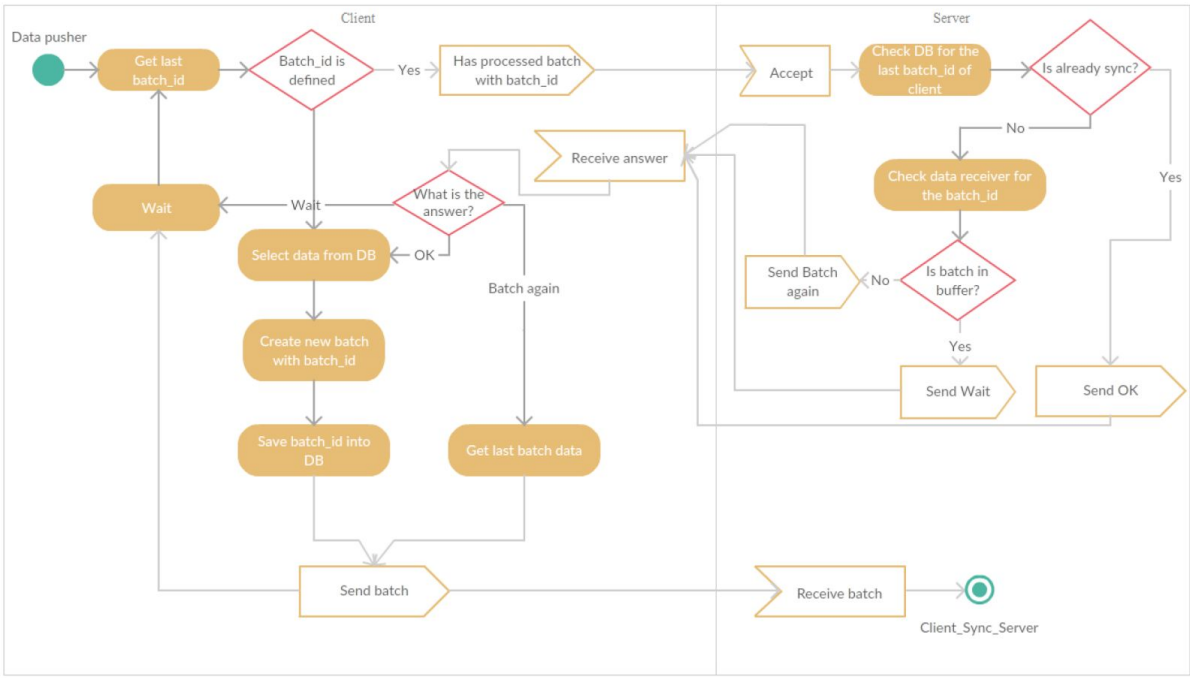
Obrázek 2: Module Diagram

Práci s webovým rozhraním znázorňuje Obrázek 3.

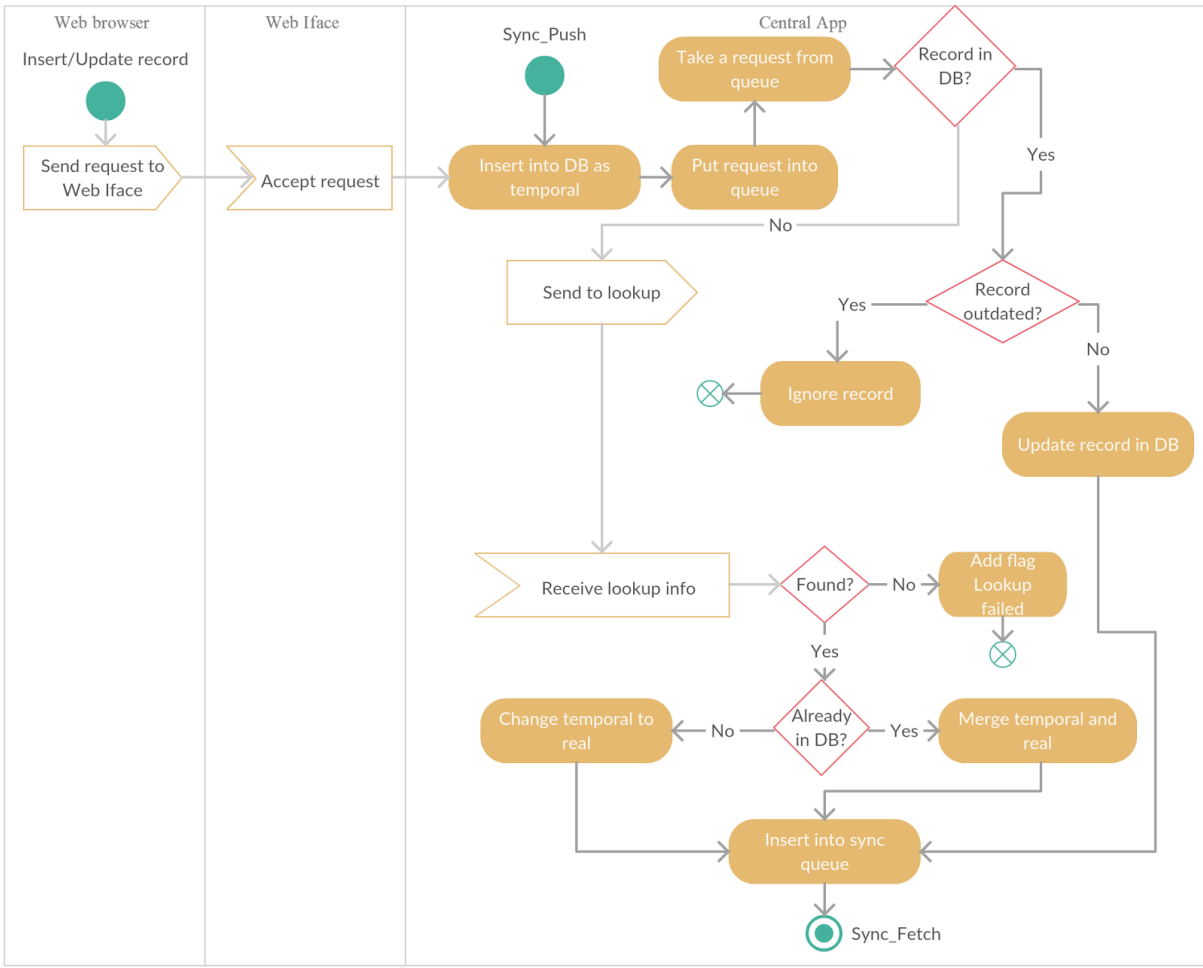


Obrázek 3: Use Case Diagram

Komunikaci mezi institucemi a centrálním serverem bude vždy inicializovat klient. Synchronizace na centrální server bude probíhat podle Obrázku 4, na který je navázán Obrázek 5.

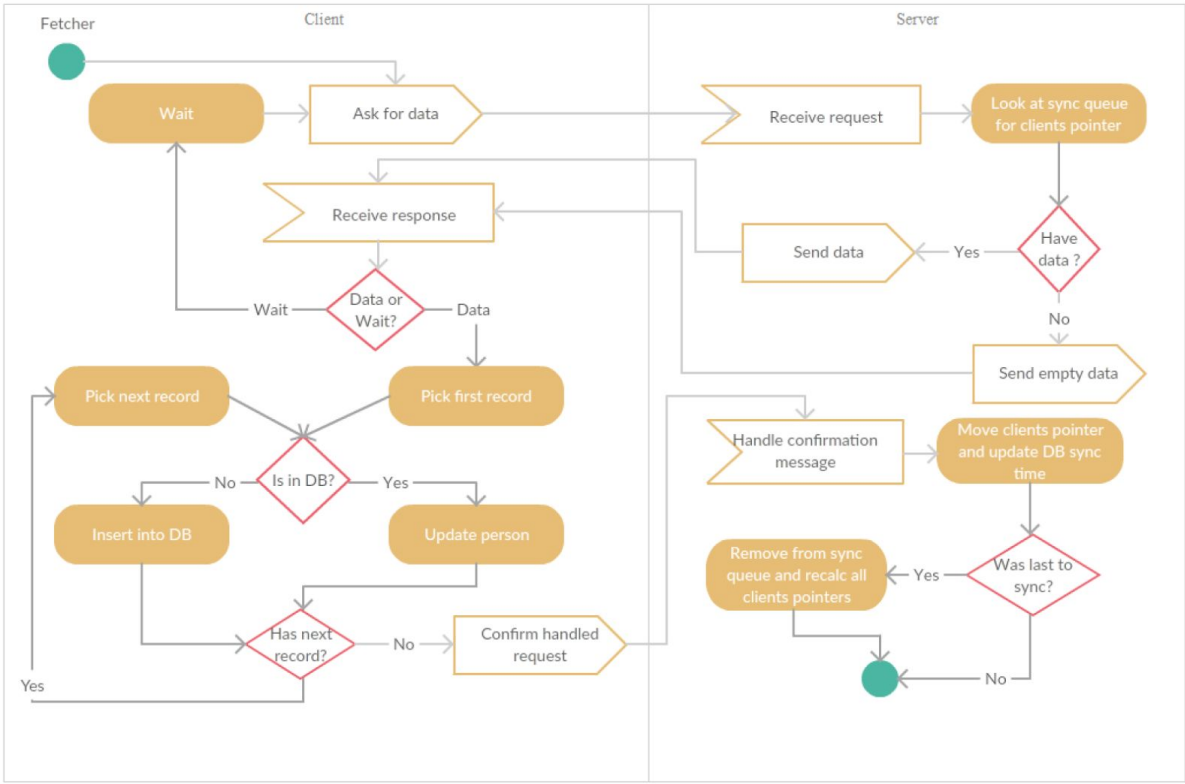


Obrázek 4: Activity Sync Push



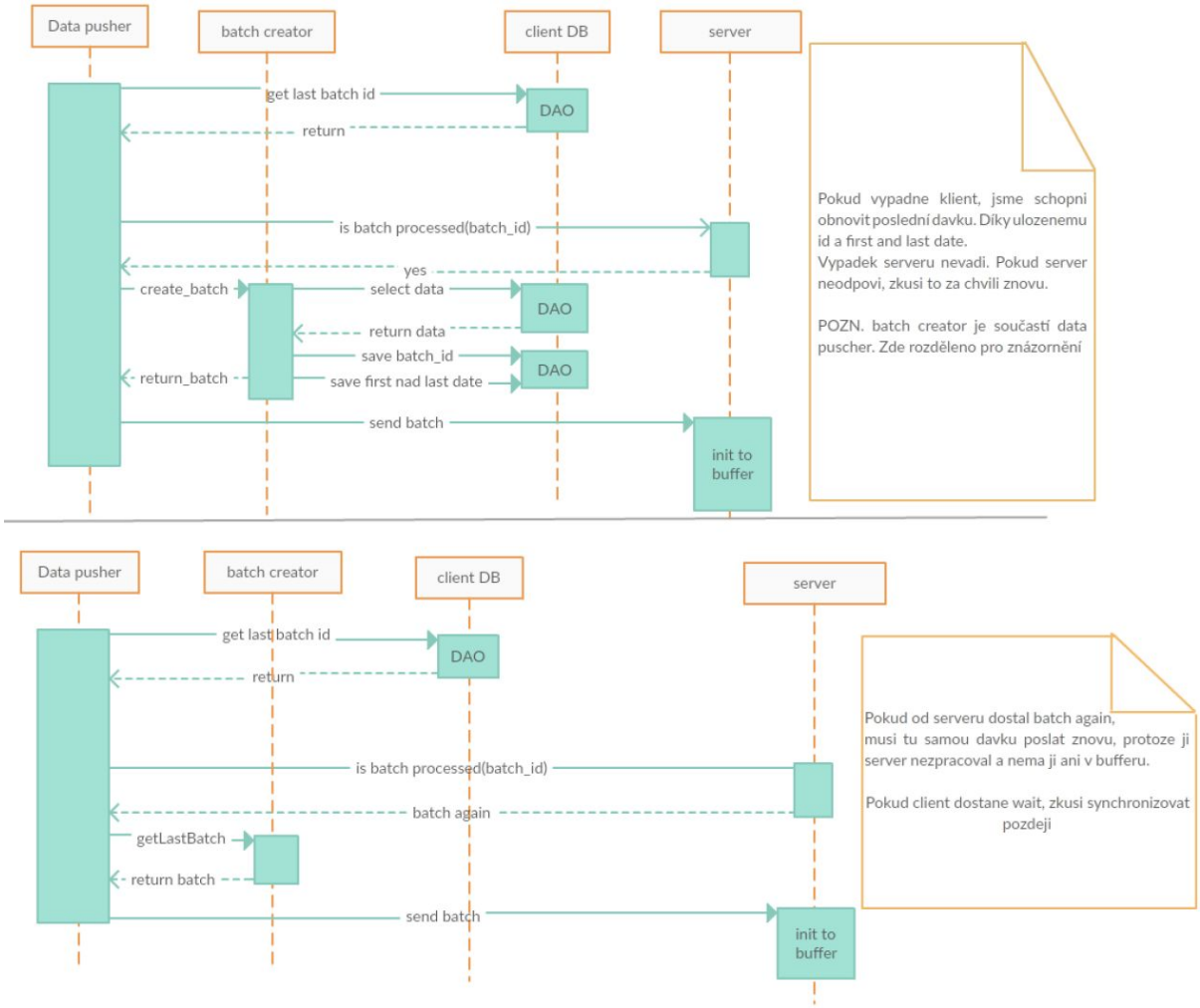
Obrázek 5: Activity WebClient Server

Synchronizace z centrálního serveru bude probíhat podle Obrázku 6, který také navazuje na Obrázek 5: Activity Client Server.

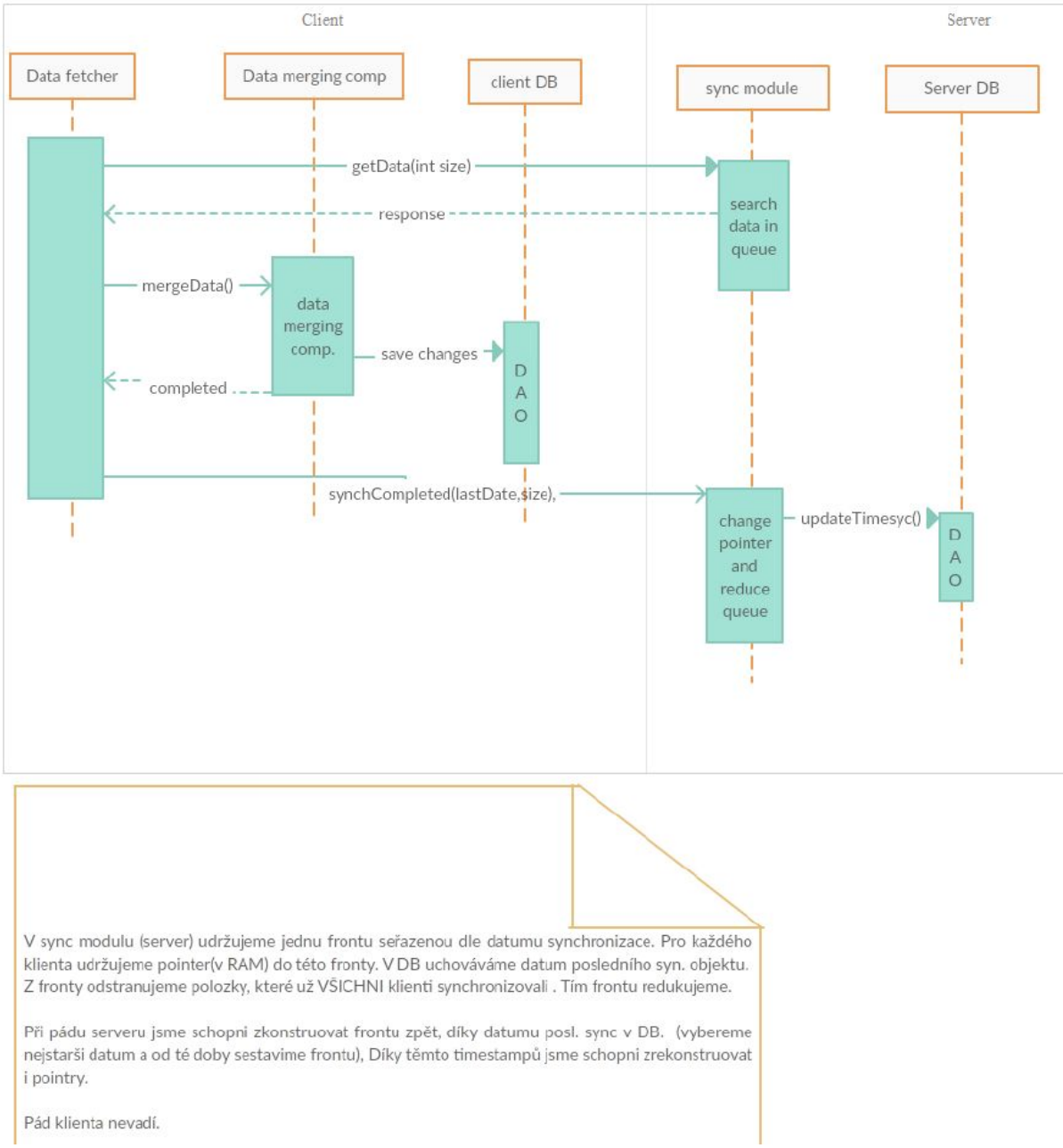


Obrázek 6: Activity Sync Fetch

Snažili jsme se myslet na možné výpadky a systém přizpůsobit tak, aby byl proti nim odolný. Proto se zde nachází potvrzování zpráv a ukládání stavů do databáze. Kritické stavy a jejich řešení jsou popsány na Obrázku 7 a Obrázku 8.

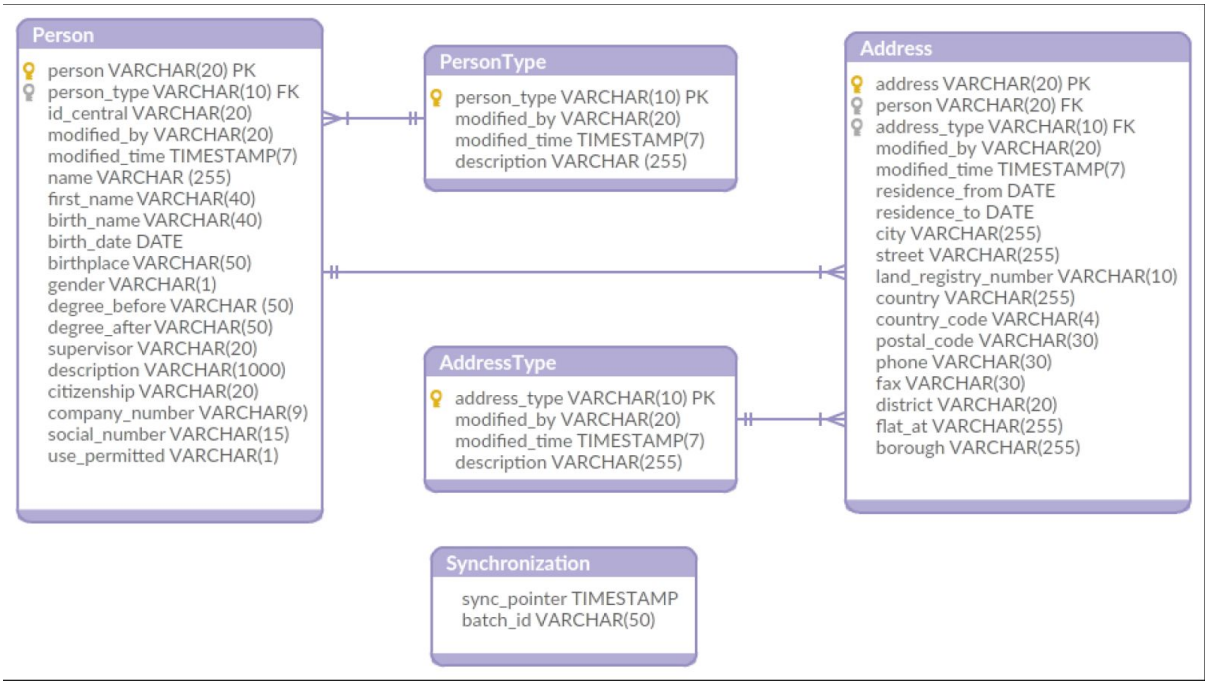


Obrázek 7: Sequence Push

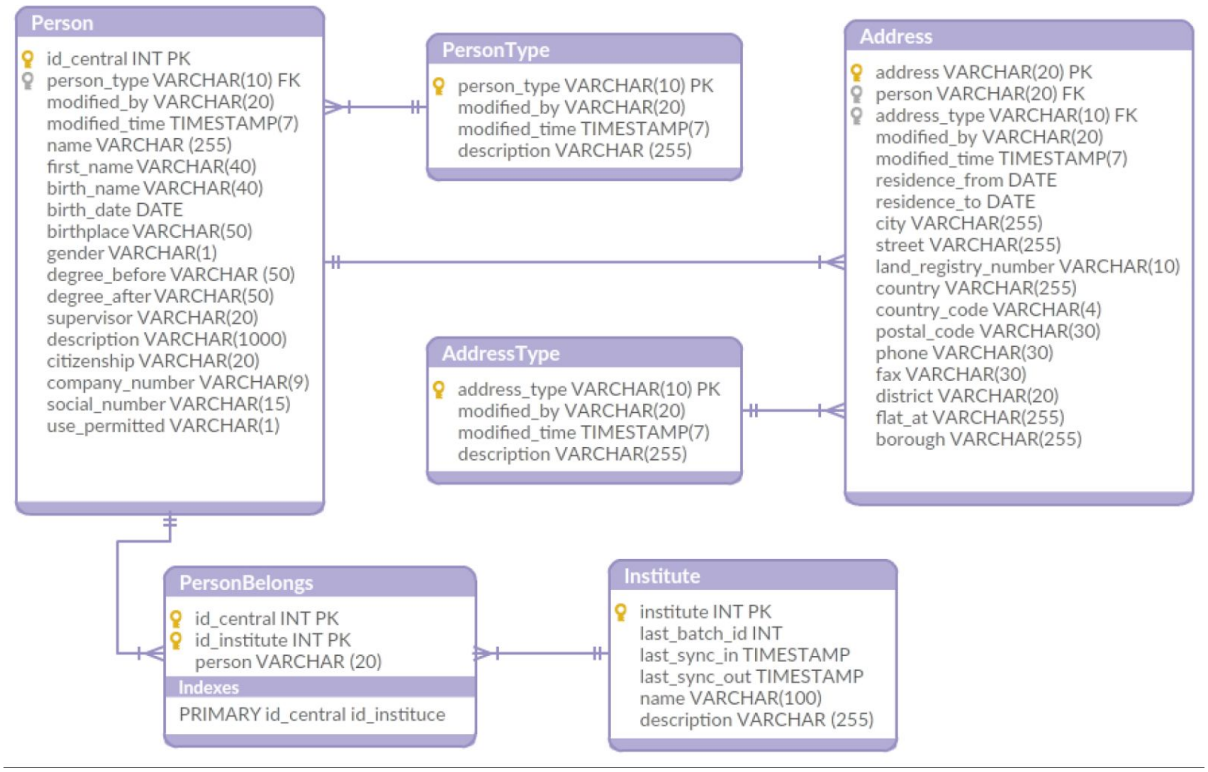


Obrázek 8: Sequence Fetch

Protože na institucích kromě aplikací CCA fungují nad databází i aplikace třetích stran a přistupují do databáze přímo, jsme omezeni na změny ve struktuře. Změny které potřebujeme se týkají přidání malého počtu sloupců do existujících tabulek (většina sloupců, které potřebujeme jíž existuje) a vytvoření nových tabulek. Z toho dostáváme dvě nová schémata pro klientskou Obrázek 9 a serverovou část Obrázek 10.



Obrázek 9: ERA Client



Obrázek 10: ERA Server

Implementace prototypu

Implementace probíhala podle návrhu.

Aplikaci tvoří tři hlavní moduly - klientský, serverový a modul common, který definuje vlastnosti a objekty, které jsou potřebné jak v klientské tak v serverové části. Kadžý z modulů je vytovřen jako maven multimodule projekt. Při sestavovaní je nejprve zapotřebí sestavit modul common až potom zbylé dva moduly.

Serverová část

Web Module

Server

Jedná se o webový server implementovaný pomocí Spring Boot. Jsou použity anotace. Autentikace je zajištěna pomocí Spring Security, která má dva uživatele s různými právy (jeden user a jeden admin). Obsahuje dva kontrolery. První je IndexController, který obsluhuje autentikaci a šablony. Druhý je ApiRestController. Ten obsluhuje RESTové rozhraní. Vyžaduje service z modulu Persistence. Jsou zde endpointy pro získání, editaci a tvorbu nových osob. Dále obsahuje endpoint pro lazy filtrování seznamu všech osob, získání číselníků a uživatelského profilu.

Webové rozhraní

Webové rozhraní je implementované jako Single Page aplikace pomocí Angularu JS. Je zde pět modulů a několik HTML stránek pro zobrazení seznamu i jednotlivých osob a jejich úpravu. První modul je App, který vlastně představuje jádro klienta a sdružuje ostatní moduly. Obsahuje směrování a inicializaci autentikace uživatele. Autentikace se řeší v modulu Auth, který v sobě drží informaci o přihlášeném uživateli a kontroluje obsluhu přihlašování a odhlašování. Další modul je Navigation, který představuje navigační lištu, využívá stavů z Auth. Dále je modul Person, který se stará o vše spojené s osobou. Obsahuje kontrolery pro seznam osob, vytváření a editaci a zobrazování. Poslední modul je Address, který se stará o přidávání, odebírání, editaci a zobrazování adres osob.

Persistence Module

Modul Persistence definuje objekty ukládané do databáze a také repository a service, které umožňují práci s nimi. Společné vlastnosti těchto objektů jsou definovány v modulu common a pomocí generičnosti jsou pouze přidány další potřebné atributy v jednotlivých implementacích (server a klient).

Person-lookup Module

Tento modul slouží k lustraci. Lustrace vyhledá stejnou osobu z databáze dle následujících kriterií:

- Hledání osoby rodného čísla a IČO

Core Module

Jádro serveru provadní samotný merge dat, obdržených z klientů. Skládá se z bufferu, mergeru a synchornizačních front.
Merge se provádí následovně:

1. Osoba z davky je už uložena v DB jako temp
2. Nalezne se persist osoba dle globalního id nebo lookup modulu
3. Pokud je nalezena persist osoba, tak je zmergována s temporal osobou. Zachovávají se údaje s pozdějším časem modifikace. Adresy jsou spojeny dohromady.
4. V případě, kdy persist osoba nalezena není, tak je uložena temporal osoba jako nová. Tato osoba má nastaven flag temporal a flag o chybě v lookup. Tento stav vyžaduje akci uživatele.

Synchronization Module

Obsahuje synchornizační frontu. Fronta je jedna pro všechny klienty. Pro každého klienta se udžuje pointer, který ukazuje do fronty. Pokud existuje nějaký prvek, za kterým jsou všechny pointery, tak je z fronty odstraněn.

Implementována je možnost, kdy každý klient má vlastní frontu. Při této variantě se bohužel uchovávají duplicitní řádky a je potřeba více paměti

Klientská část

Pusher

Pusher slouzi k posilání změn z klienta na server. Princip je znázorňen na prislušných diagramech.
Důležitým algoritmem je postup při vytváření dávky. Ta se vytváří tak, že se vybírají změny, které spadají do daného časového intervalu modifikace. Pokud buffer pro dávku není plný, tak se zmíněný interval zvětší (v reálu se posune, aby se nevybíraly osoby již vybrané). Zvětšuje se do té doby dokud se nepřekročí povolená velikost bufferu. Důsledkem tohoto algoritmu je, že uvedená velikost bufferu znázorňuje minimální počet poslaných prvků. Velikost kroku změny intervalu lze nastavit v properties souboru.
Pro předávání dat slouží wrapper, který obsahuje společné atributy pro klientskou a serverovou část. Data je nutné před posláním normalizovat do příslušné podoby.

Fetcher

Fetcher je komponenta sloužící pro synchronizaci dat ze serveru. Fetcher pošle REST request na server, ve kterém je obsažen identifikátor příslušného klienta a velikost dávky, kterou chce přijmout. Tato velikost je pouze minimální viz předchozí odstavec. Fetcher přijme od serveru data a předá je mergeru. Po provedení mergeru, musí serveru potvrdit, že danou dávku zpracoval.

Opět pro přenos dat slouží wrapper. Proto přijatá data je potřeba převést zpět z normalizované podoby.

Merger

Merger slouží k mergování dat. Klientská id jsou unikátní pouze vůči lokální db. Nejsou unikátní v rámci centralizovaného systému. Merger se proto snaží nalézt stejnou osobu, kterou se snažíme zmergovat. K tomu používá:

- Lokální id, které se může uchovávat na serveru.
- Hledání osoby podle globáního id (id dle serveru). Globalni id je posíláno ze serveru a na clientu je ukládáno.
- Hledání osoby rodného čísla a iča

Pokud je nalezena odpovídající osoba, tak je osoba zmergována. Zachovávají se údaje s pozdějším časem modifikace. Adresy jsou spojeny dohromady.
V případě, kdy příslušná osoba nalezena není, tak je uložena jako osoba nová.

Persistence Module

Slouží k namapování databáze a k práci s ní. Dao vrstva je tvořena pomocí springovské Crud Repository a příslišných services.

Uživatelská dokumentace

Client main

Client se díky Spring Boot spouští jako obyčejná aplikace pomocí třídy: Application. Ta je umístěna v modulu core a obsahuje metodu main().
Před spuštěním klienta už musí běžet server.

Client properties soubor

Thread.pusher.timout - znázorňuje jak často se mají posílat data na server
Thread.fetcher.timout - znázorňuje jak často se mají data stahovat ze serveru
Client.id - id clienta
Spring.datasource - nastaveni databaze
Uri - uri na rest
batchCreator.step - skok zvětšení časového intervalu
batchCreator.bufferSize - velikost bufferu pro push
startTime - začateční čas pro push.
fetch.bufferSize - velikost bufferu pro fetch
Zde je přiklad nastavení

```
#thread timeouts
thread.pusher.timeout =1000
thread.fetcher.timeout =1000
#Rest config
client.id =0
```

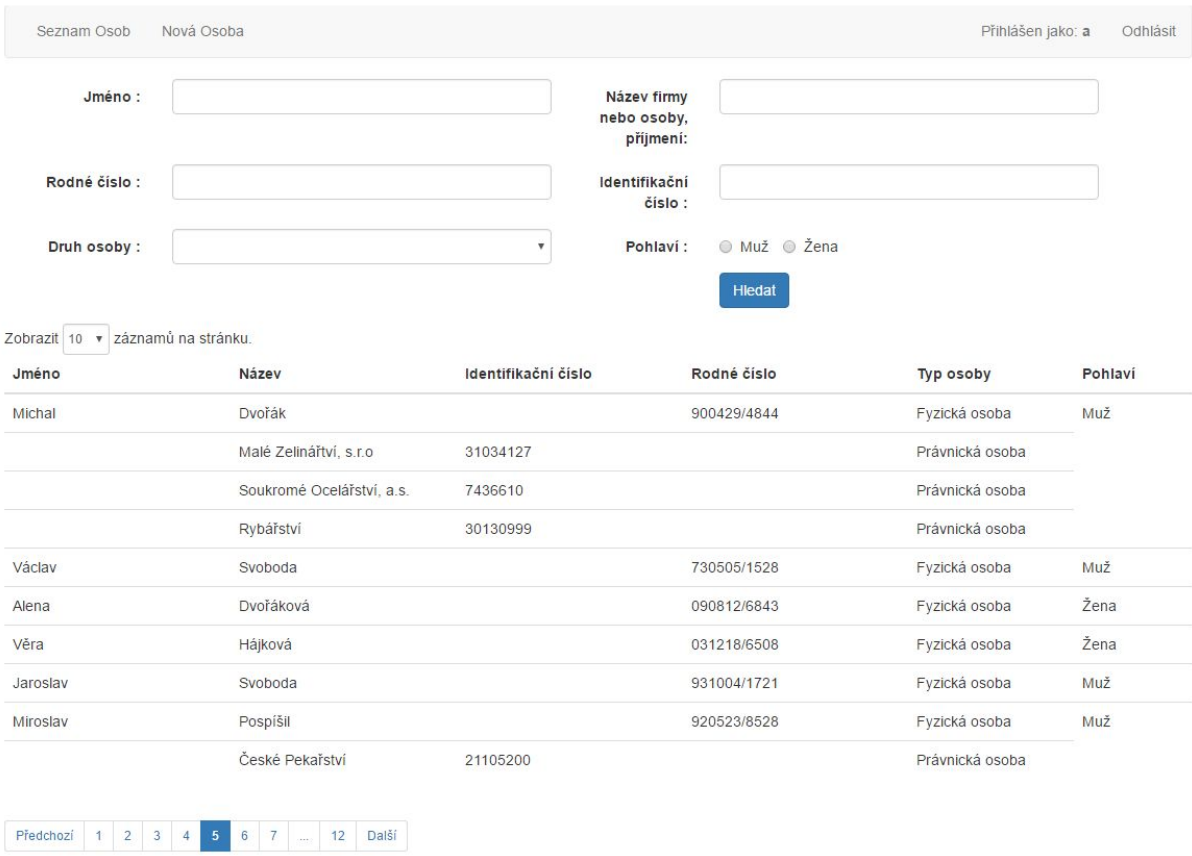
```
uri.fetch.confirm=http://localhost:8081/sync/data/confirm
uri.fetch=http://localhost:8081/sync/data/fetch
uri.getData=http://localhost:8081/sync/data
uri.lastBatch=http://localhost:8081/sync/lastBatch
# batch creator - time in ms
batchCreator.step=3600000
batchCreator.bufferSize=100
# fetcher
fetch.batchSize=100
#pusher
startTime=0
```

Server properties soubor

Obsahuje pouze nastavení kontextu aplikace a portu na kterém běží. Dále se zde nastavuje spojení s databází.

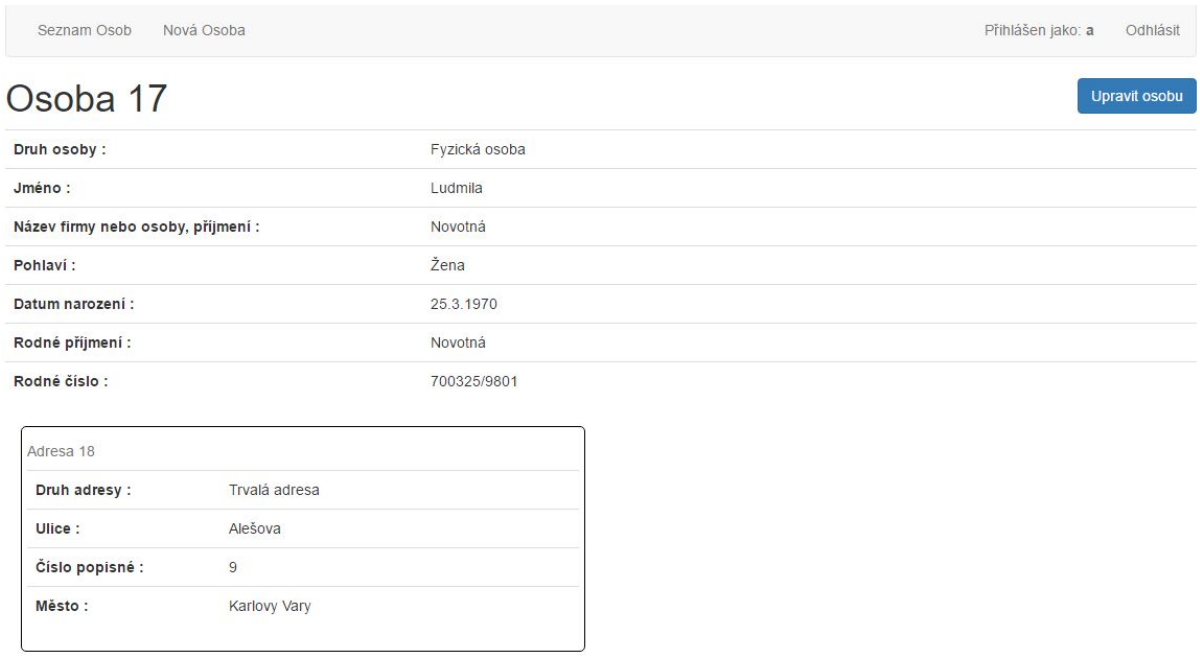
Webové rozhraní

Uživatelé:
Obyčejný s rolí USER
Login: u
Heslo: u
Editační s rolí ADMIN
Login: a
Heslo: a
Ovládání je intuitivní.
Seznam osob, který je na Obrázku 11, obsahuje základní filtry pro lehčí orientaci a je zobrazen pomocí stránek o volitelném počtu záznamů. Kliknutím na příslušný řádek s osobou se přejde na stránku s jejím detailem.



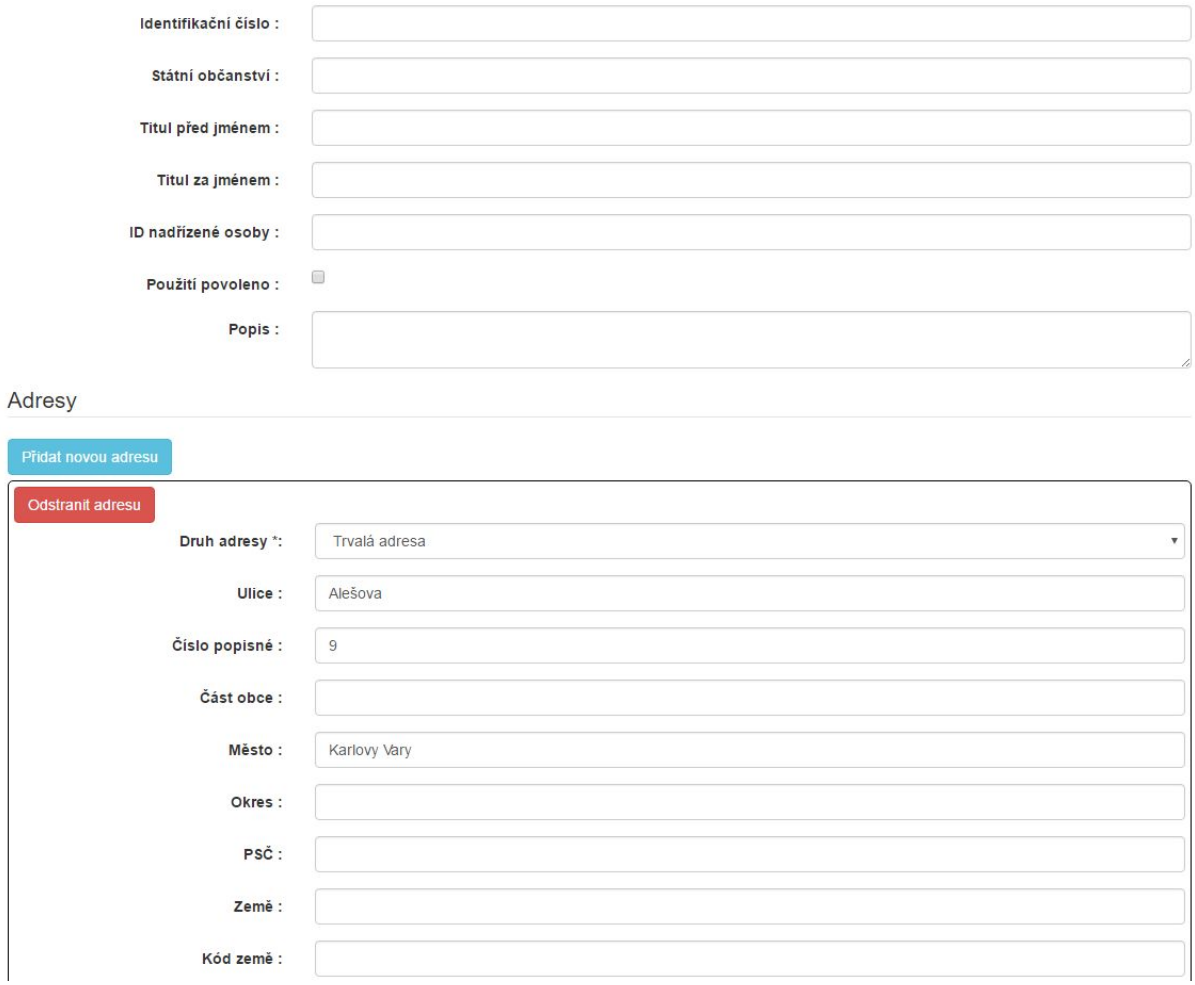
Obrázek 11: Seznam Osob

Při zobrazení detailu osoby, který je Obrázku 12, se zobrazují pouze vyplněné parametry a adresy. Pokud je uživatel admin, zobrazí se mu tlačítko pro přechod do editačního formuláře.



Obrázek 12: Detail osoby

Úprava osoby nebo založení nové je v formuláři, který je vidět na Obrázku 13. Pokud uživatel přišel z existující osoby, hodnoty jsou předvyplněné. Lze zde také přidávat, upravovat a odebírat adresy.



Obrázek 13: Úprava osoby

Závěr

Cílem práce bylo navrhnout a implementovat architekturu, která zcentralizuje daný systém. Dalším cílem bylo vytvořit jednoduchou webovou aplikaci. Všechny tyto cíle se podařilo splnit v celém svém rozsahu.

Pří návrhu architektury se nám zdálo zbytečné vizualizovat každý speciální stav. Nicméně při samotné implementaci nám to ulehčilo mnoho starostí a komunikace mezi členy týmu.