Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Off-line portál pro správu elektrofyziologických experimentů

Plzeň, 2014 Michal Kasal

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května

Michal Kasal

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Romanu Moučkovi, Ph. D. za jeho odborné rady. Také bych chtěl poděkovat Ing. Petru Ježkovi, Ph. D. za pomoc s řešením problémů s Portálem. V neposlední řadě bych také chtěl poděkovat všem testerům, kteří si našli čas a otestovali výslednou aplikaci.

Abstract

The EEG/ERP Portal is a web application, which is used for storing, sharing and managing data and metadata from electrophysiological experiments. It is developed at the Department of Computer Science and Engineering. An offline EEG/ERP Portal client was created to store experimental data at the locations where Internet connection is not available. However, it was developed for an old database model which is not currently used.

This bachelor thesis focuses on analysis of the currently used database model in a new version of the EEG/ERP Portal. The aims of this paper are to create a new data model used by the offline client, explore synchronization models and approaches, choose an appropriate synchronization solution, which allows proper data synchronization between the client and the Portal, and update the offline client to work with the new database model.

Keywords

Synchronization, EEG/ERP Portal, SymmetricDS, relational database, neuroinformatics, offline client

Abstrakt

EEG/ERP Portál – webová aplikace, která umožňuje ukládat, sdílet a spravovat data a metadata elektrofyziologických experimentů, je vyvíjena na Katedře informatiky a výpočetní techniky. Pro EEG/ERP Portál byl vytvořen off-line klient, který umožňuje ukládat experimentální data bez připojení k internetu. Bohužel byl vyvinut pro starý databázový model, který se dnes nepoužívá.

Tato bakalářská práce se zaměřuje na analýzu současně používaného databázového modelu používaného v novém EEG/ERP Portálu. Cílem této práce je vytvoření nového datového modelu používaného klientem, prozkoumat synchronizační modely a přístupy a vybrat vhodné synchronizační řešení, které by umožnilo řádnou synchronizaci dat mezi klientem a Portálem, a aktualizovat off-line klienta, aby byl schopen pracovat s novým databázovým modelem.

Klíčová slova

Synchronizace, EEG/ERP Portál, SymmetricDS, relační databáze, neuroinformatika, off-line klient

Obsah

1 Úvod

Na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni je vyvíjena webová aplikace sloužící k uchovávání, sdílení a správě dat neuroinfromatických experimentů, nazvaná EEG/ERP (Electroencephalography/event-related potentials) Portál. Protože experimenty mohou probíhat v místech, kde se vyskytují problémy s internetovým připojením, byl vytvořen off-line klient, který umožňuje ukládat naměřená data a ta poté synchronizovat s Portálem.

EEG/ERP Portál se však v průběhu času vyvíjel a došlo k některým zásadním úpravám. Změnil se datový model i použitá databázová platforma. Se změnami Portálu je nutné provést změny off-line klienta. Cílem této práce je analyzovat současný stav řešení, navrhnout nový datový model odpovídající datovému modelu EEG/ERP Portálu a vyřešit problémy se synchronizací mezi klientem a Portálem, které obsahovala předchozí verze.

Tato práce nejprve vysvětluje základní pojmy v neuroinformatice. Další část je věnována analýze EEG/ERP Portálu a off-line klienta. Poté se práce věnuje synchronizačním modelům a přístupům a také problémům, které mohou během synchronizace nastat. Následuje přehled některých synchronizačních nástrojů a jejich srovnání. Na tu kapitolu navazuje podrobnější popis zvoleného synchronizačního nástroje, po kterém se práce zabývá změnami provedenými v off-line aplikaci a popisuje její finální strukturu. Poslední část práce je věnována testování aplikace.

2 Neuroinformatika

Neuroinformatika je vědní obor, který se zabývá analýzou, ukládáním, zpracováním a sdílením heterogenních dat, které generují neurovědy. Zkoumá, jakým způsobem lze organizovat data tak, aby je bylo možné snadno sdílet, prohledávat, analyzovat a vytvářet z nich realistické modely. Cílem neuroinformatiky je porozumění činnosti mozku a zjištění příčin jeho poruch a s nimi souvisejících onemocnění [3].

2.1 Elektroencefalografie

Elektroencefalografie (EEG) je jednou z funkčních vyšetřovacích metod, která umožňuje měření a záznam aktivity mozku. Před měřením je nanesen konduktivní gel, který snižuje elektrický odpor. Na skalp se připevní systém elektrod nebo síť s napevno přidělanými elektrodami, jejichž umístění je zpravidla dáno Mezinárodním systémem rozložení elektrod 10-20. Elektrody zaznamenávají oscilace elektrického napětí, které vzniká aktivitou neuronů převážně v šedé kůře mozkové [19].

Elektroencefalografie představuje rychlou neinvazivní metodu, která poskytuje základní informace o mozkové aktivitě, díky které lze například zkoumat epilepsii nebo úroveň bdělosti. Možnosti EEG jsou ovšem limitované, protože principiálně umožňuje zkoumat aktivitu mozku jako celku, nikoliv na úrovni jednotlivých kognitivních procesů [19].

2.2 Kognitivně evokované potenciály

Evokované potenciály (EP) jsou odpovědi nervové soustavy na stimulaci receptorů například světelnými záblesky, tóny, elektrickými šoky nebo stiskem. Kognitivně evokované potenciály (ERP) se řadí mezi EP. Jedná se o podněty vázané na konkrétní identifikovatelnou událost a oproti jiným evokovaným potenciálům nastávají déle od stimulu [1].

3 EEG/ERP portál

EEG/ERP portál je webová aplikace (viz obr. 1), která umožňuje zadávat, ukládat a sdílet data experimentů, které souvisejí s měřením mozkové aktivity. Systém sjednocuje různé druhy dat a umožňuje k nim přistupovat přes webové rozhraní.

Obrázek 1: Úvodní stránka EEG/ERP portálu [28]

3.1 Funkce EEG/ERP portálu

Mezi základní funkce Portálu patří:

- Autentifikace uživatele.
- Ukládání, aktualizace a stahování
  - experimentálních dat,
  - metadat experimentů,
  - scénářů experimentů,
  - informací o testovaných osobách.
- Fultextové vyhledávání.
- Práce ve výzkumných skupinách, rozdělení rolí uživatelů.
- Publikování článků a novinek.
- Přihlášení pomocí Facebook.com a LinkedIn.com.

3.2 Architektura EEG/ERP portálu

Portál má vrstvenou architekturu.

3.2.1 Datová vrstva

Portál uchovává část aplikačních dat v relační databázi a část v nerelační. Pro uchování dat v relační databázi je použita databázová platforma PostgreSQL. Přístup k databázi je zprostředkován pomocí objektově relačního mapování, ke kterému Portál používá framework Hibernate. Hibernate mapuje tabulky relační databáze k objektům POJO (Plain Old Java Object), které obsahují definice objektů a mapovací parametry určené anotacemi. Přístup k POJO objektům zajišťují třídy DAO (Data Access Object) [14]. V relační části jsou uloženy všechna administrativní data Portálu, například informace o výzkumných skupinách.

Nerelační část je tvořena JSON (JavaScript Object Notation) objekty, které jsou uchovávány v nerelační databázi ElasticSearch [8]. V nerelační části jsou uloženy nejrůznější parametry experimentů a osob, například jsou v ní uloženy informace o použitém hardwaru a softwaru.

3.2.2 Aplikační vrstva

Aplikační vrstva využívá Spring Framework s integrovanými moduly Spring Core, MVC, Security a AOP (Aspect Oriented Programming). Uživatelské akce jsou ověřovány validátorem, o který se starají třídy controller [18].

3.2.3 Prezentační vrstva

Portál je dostupný přes libovolný webový prohlížeč bez potřeby instalace dodatečných nástrojů. K tomuto účelu aplikace využívá Apache Wicket, což je framework prezentační vrstvy, který umožňuje oddělit HTML (HyperText Markup Language) od kódu psaného v Javě [5].

3.3 Webová služba pro komunikaci s klientem

Portál obsahuje webovou službu pro komunikaci s klientem, která umožňuje autentifikaci uživatele, stažení a nahrání některých dat klientem. Jedná se o třídu ClientService, která obsahuje metody pro ukládání konkrétních záznamů do databáze Portálu a metody pro vracení seznamů s obálkami požadovaných záznamů. Třída ke komunikaci používá protokol SOAP (Simple Object Access Protocol), který používá jako zprávy XML dokumenty [14].

3.4 Vývoj Portálu v čase

V následující podkapitole je stručně popsán vývoj Portálu od verze, pro kterou byl původně naimplementován off-line klient. V té době používal EEG/ERP Portál pouze relační databázi. Jako databázová platforma byla zvolena Oracle 11g. Synchronizace proto byla testována nejdříve na této platformě a původním datovém modelu. Stejně jako konečný prototyp vytvořený v předmětu Projekt 5. Také prezentační vrstva Portálu byla jiná. Pro zobrazení stránek do standardních HTML byla používána technologie JSP (Java Server Pages), kam byla vkládána data určená k zobrazení pomocí technologie JSTL (JavaServer Pages Standard Tag Library) [18].

Migrace databáze Portálu na databázovou platformu PostgreSQL přišla až s verzí 2.0, která byla vydána 22. ledna 2014. Portál také přešel na používání Wicket Frameworku v prezentační vrstvě. V září roku 2013 byla zadána práce, která měla zjistit, jak by se dala část databáze převést do nerelační části. Jako nerelační databáze je používán Elastic Search. Poslední změna souborů, které se starají o ukládání dat do nerelační části, je z konce února roku 2014. V té době se data ukládají jak do relační, tak redundantně do nerelační části. Toto je poslední verze portálu k datu napsání této práce. Z důvodu souběžného vývoje EEG/ERP Portálu a off-line klienta tak došlo k implementaci klienta na verzi 2.0 vydanou na konci ledna roku 2014.

4 Analýza původní aplikace

Původní verze off-line klienta byla vytvořena v rámci diplomové práce Ing. Františka Lišky [14].

Off-line klient pro EEG/ERP portál umožňuje prohlížet a přidávat data související s výzkumem elektrofyziologických experimentů. Aplikace je navržena jako tlustý klient a je funkční i bez připojení k internetu. Provedené změny umožňuje ukládat do své lokální databáze a pak je dávkově nahrát na server.

4.1 Funkce původního řešení

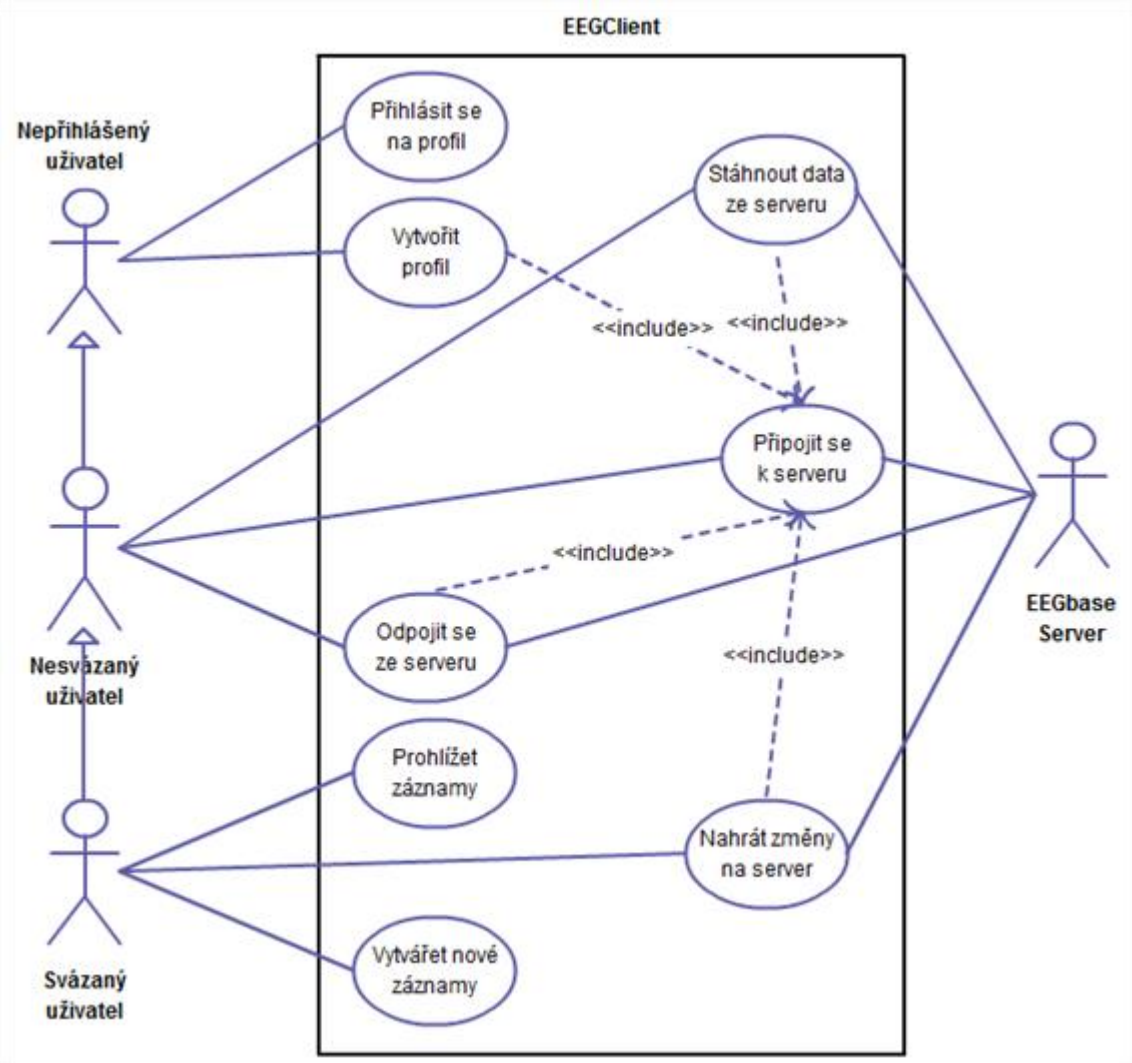
Původní řešení poskytuje základní funkce pro přidávání entit a jejich zobrazování.

Umožňuje:

- ověřit přihlášení
- omezenou synchronizaci
    - stažení dat (svázání profilu)
    - nahrání nových dat na server
- přepínání mezi on-line a off-line režimem
- zobrazení a přidávání
    - experimentů
    - scénářů
    - výzkumných skupin
    - osob
    - zařízení
    - počasí
    - parametrů osob
    - parametrů experimentu
    - metadat souborů

Funkčnost aplikace se liší podle typu uživatele. Aplikace rozpoznává tři typy uživatelů: nepřihlášeného uživatele, nesvázaného uživatele a svázaného uživatele. Svázáním se zde rozumí proces, ve kterém se klient připojí k Portálu, který provede ověření přihlašovacích údajů. Po úspěšném ověření je klientem inicializováno stažení počátečních dat z databáze Portálu. Počátečními daty se rozumí výzkumné skupiny, scénáře, osoby a seznam nadefinovaných parametrů. Po dokončení stažení dat je uživatel svázán s Portálem [14].

Svázaný uživatel může zobrazovat neuroinformatická data bez připojení k Portálu. Funkce poskytované aplikací vzhledem k jednotlivým typům uživatele jsou vyobrazeny v diagramu případů užití (obr. 2).

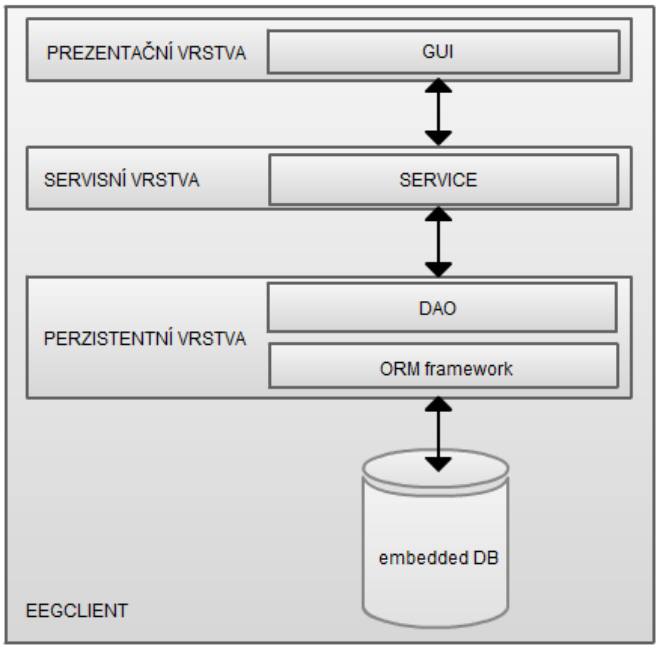


Obrázek 2: Diagram případů užití [14]

V původní aplikaci je svázání uživatele provedeno prostřednictvím metody linkPerson třídy DataService. Informace o svázání uživatele s Portálem se jako příznak zaznamená do tabulky PROFILE, kde je uložen identifikátor osoby z tabulky PERSON lokální databáze, stejně jako je v ní uložen identifikátor osoby v tabulce PERSON databáze Portálu [14].

4.2 Architektura aplikace

Aplikace využívá vrstvenou architekturu, která se vyznačuje rozdělením aplikace do vrstev s různými funkcemi. Vrstvy komunikují pouze s vrstvami, které se nacházejí v hierarchii bezprostředně nad nimi nebo pod nimi. Na ostatních vrstvách jsou nezávislé [16]. Architektura aplikace je znázorněna na obrázku 3.



Obrázek 3: Vrstvená architektura aplikace [14]

4.2.1 Prezentační vrstva

Prezentační vrstva slouží ke komunikaci uživatele s programem. Je tvořena grafickým rozhraním, využívajícím mimo jiné i vlastní grafické komponenty. Uživatelské rozhraní je implementováno pomocí knihoven Swing [14].

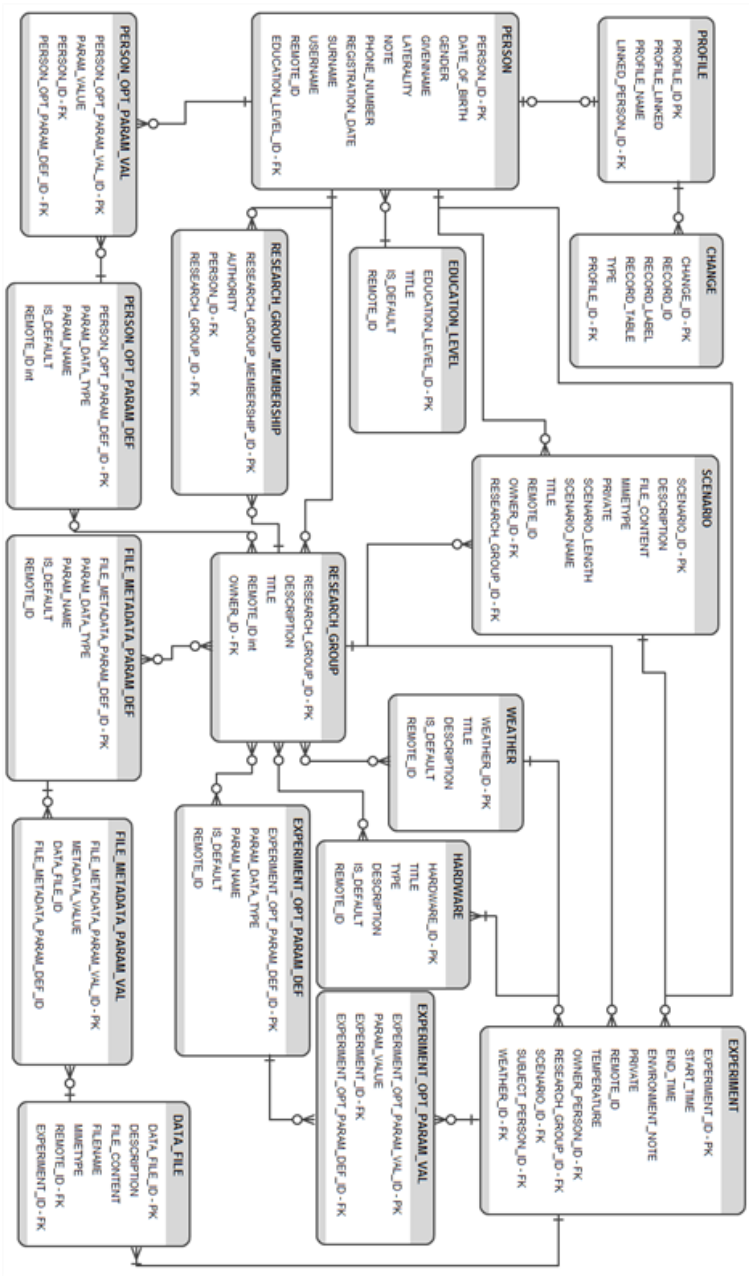
4.2.2 Servisní vrstva

Servisní vrstva původní aplikace slouží pro komunikaci mezi datovou a prezentační vrstvou. Jsou v ní obsaženy třídy, které zprostředkovávají stažení a nahrání dat na Portál a svázání uživatele s Portálem.

4.2.3 Perzistentní vrstva

Perzistentní vrstva zprostředkovává rozhraní mezi servisní vrstvou a databází. Jednotlivé tabulky databáze jsou na objektové úrovni reprezentovány jako POJO objekty, ke kterým se přistupuje přes pomocné objekty DAO [14].

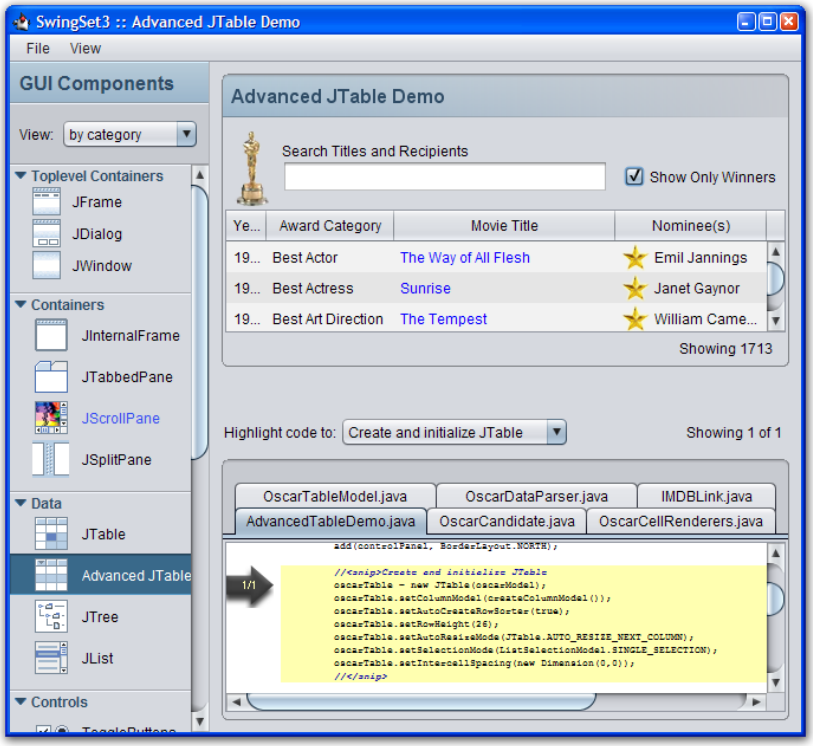
Klient využívá pro ukládání dat jednouživatelskou embedded databázi Apache Derby. Původní datový model databáze je vyobrazen na obrázku 4.



Obrázek 4: ERA diagram původního datového modelu off-line klienta [14]

4.3 Použité prostředky původní aplikace

Aplikace je naprogramována v programovacím jazyce Java. Pro její funkčnost je z důvodu použití vzhledu Nimbus požadována minimálně verze 6 s updatem 10. Nimbus je multiplatformní vizuální styl (viz obr. 5), který využívá vektorovou grafiku místo statických bitmap. [21]



Obrázek 5: Ukázka vzhledu Nimbus [21]

K řešení závislostí na externích knihovnách a automatizovanému sestavení programu je použit Apache Maven 4. Generování kódu tříd webových služeb zajišťuje knihovna Apache CXF 2.7. Aplikace používá framework pro objektově relační mapování Hibernate 3.6. Ten umožňuje například vytvářet novou databázi, pokud neexistuje, po spuštění klienta [14]. Většina POJO a DAO objektů je převzata z EEG/ERP portálu.

Klient je vystavěn podobně jako Portál na frameworku Spring s využitím návrhového vzoru Inversion of Control. V původním řešení je použita verze 3.2.2. Klient pro uložení dat využívá embedded databázi Apache Derby ve verzi 10.10, která umožňuje jednouživatelský přístup k databázi. [14]

4.4 Řešení synchronizace dat

Synchronizace klienta je řešena v servisní vrstvě třídami DownloadService, UploadService a WebServiceClient. WebServiceClient ověřuje, zda je možné se připojit k webové službě Portálu a poté provádí samotné spojení. DownloadService stahuje data z Portálu a UploadService je tam nahrává. V původním klientovi nebylo možné stáhnout již proběhlé experimenty.

Nahrávání změn probíhá s pomocí tabulky CHANGE v databázi klienta, která obsahuje záznamy, které byly klientem přidány. Při úspěšném odeslání záznamu je tento záznam smazán z tabulky a v příslušné tabulce záznamu je nastaveno jeho REMOTE_ID, které koresponduje s identifikátorem přiděleným Portálem [14]. V původní verzi nelze nahrát všechny nové záznamy a také není možné záznamy měnit, ani odstraňovat.

Toto řešení není optimální. Jednak vzniká nepřiměřená zátěž na centrální uzel sítě, který musí každý záznam vkládat s jiným ID, zaznamenat změny a odesílat je zpět příslušným klientům a jednak je třeba v souvisejících záznamech aktualizovat cizí klíče, aby nedošlo k porušení referenční integrity. Synchronizace databáze, která obsahuje značné množství rozkladových tabulek, se tak stává složitou s vysokými nároky na výkon databázového serveru Portálu.

Protože je synchronizace dat nezbytnou základní funkcí off-line klienta, je na ní tato práce z velké části zaměřena. V následující kapitole jsou popsány nejdůležitější vlastnosti a funkce vybraného řešení, stejně jako princip, podle kterého jsou data synchronizována.

5 Synchronizace

Základní funkcí off-line klienta je schopnost synchronizace dat, což je proces zachování konzistentních a uniformních dat ve všech databázích sítě. Portál a klienti jsou uspořádány v síti s hvězdicovou topologií, kde klienti komunikují výhradně s Portálem a jeho prostřednictvím. Hvězdicové uspořádání přináší nevýhodu v takzvaném jediném bodu selhání (single point of failure), kterým je centrální uzel – Portál. V případě výpadku centrálního uzlu dojde k přerušení možnosti synchronizace dat a tím, pokud klienti ještě nedisponují daty, také k nemožnosti jejich používání. Je tedy nutné minimalizovat toto riziko návrhem synchronizačního řešení, které je odolné vůči chybám, například neočekávanému výpadku internetového připojení. Možným řešením je použití redundantních databázových serverů.

5.1 Synchronizační modely

5.1.1 Master-slave model

V hvězdicové topologii se nabízí použití modelu synchronizace označovaného jako master-slave (někdy jako primary copy) [23]. V tomto modelu vystupuje Portál jako hlavní uzel (master), který propaguje změny ke klientům, což jsou v tomto modelu podřízené uzly (slaves). Klienti data přijímají a ukládají do své lokální databáze. Konflikty v databázi Portálu nenastávají a data v ní zůstávají konzistentní. V případě změn v databázích klientů, pak může docházet ke konfliktům, které jsou však řešeny na úrovních jednotlivých klientů a nijak neovlivňují databázi centrálního uzlu. Umožnění těchto zápisů klientů však vede k divergenci dat. Tento synchronizační model lze použít pouze za předpokladu, že klienti nemohou změny v datech propagovat do centrální databáze Portálu [23].

5.1.2 Multi-master model

Off-line klient je ale navržen tak, aby bylo možné data upravovat a provedené změny odesílat do databáze Portálu a dále jiným klientům. Jediným možným synchronizačním modelem, který proto lze aplikovat, je takzvaný multi-master model, také nazýván jako update everywhere model [23]. V multi-master modelu vystupují Portál a klienti jako rovnocenné master uzly, které mohou vkládat, měnit a mazat data svých databází a tyto změny obousměrně propagovat do ostatních uzlů sítě. To s sebou přináší nutnost detekovat a řešit případné kolize v datech nebo jim předcházet.

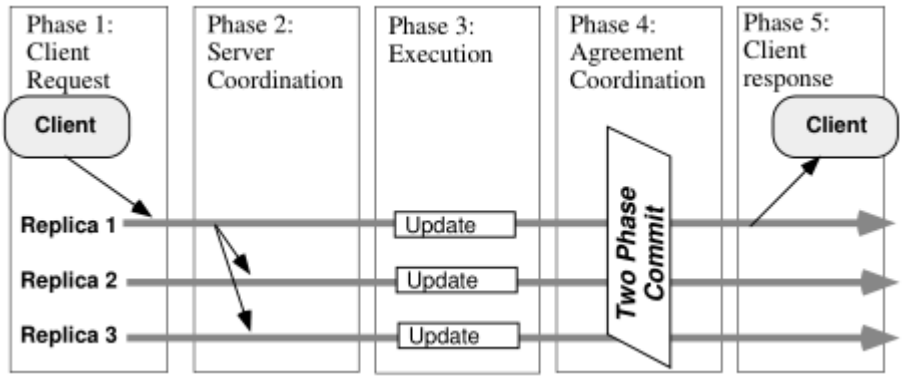
5.2 Synchronizační přístupy

5.2.1 Synchronní způsob synchronizace

Synchronní synchronizace (eager synchronization) je způsob synchronizace, který umožňuje předcházet konfliktům v datech. Databáze ostatních uzlů jsou aktualizovány v rámci původní transakce, která je ukončena až tehdy, když jsou všechny databáze převedeny do stejného stavu. Pořadí operací a předcházení konfliktů určuje ten, kdo synchronizaci vyvolal, ostatní uzly se podřizují. Tím je zajištěno, že všechny aplikace používají aktuální data a jsou konzistentní [27].

Jednou z technik, kterou lze docílit synchronní synchronizaci, je metoda distribuovaných zámků (viz obr. 6), kdy před zahájením synchronizace dojde k odeslání požadavku na uzamčení záznamu v ostatních uzlech [27]. Teprve když jsou všechny uzly zkoordinovány, může být vykonána požadovaná operace.

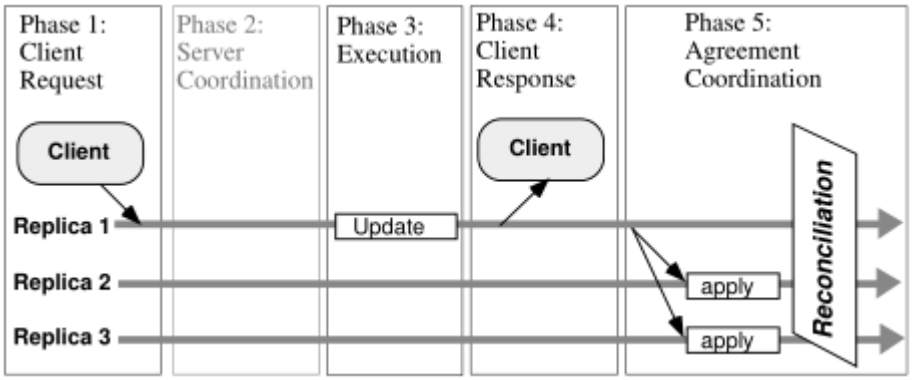
Z podstaty synchronního způsobu vyplývá, že jej nelze v off-line klientovi aplikovat, neboť by synchronizace proběhla až v době, kdy by byly ostatní uzly připojené k síti. Do té doby by došlo k uzamčení záznamů a aplikaci by nebylo možné používat. Proto se synchronní způsob používá zpravidla jen tam, kde jsou uzly neustále připojené k síti.



Obrázek 6: Eager update everywhere synchronization s využitím zámků [27]

5.2.2 Asynchronní způsob synchronizace

Druhou možností je použití asynchronní synchronizace (označované také jako lazy synchronization), během které je nejprve změněna lokální databáze a k propagaci změn dochází až po nějakém čase, kdy už je původní transakce ukončena (viz obr. 7). Změny ostatních databází tak probíhají v rámci rozdílných transakcí v různou dobu [27]. Nelze vyloučit, že někteří klienti proto budou používat neaktuální data. U asynchronního přístupu kombinovaného s multi-master modelem mohou vznikat konflikty v datech, kterým nelze předcházet, ve všech uzlech sítě [23, 27]. Z tohoto důvodu je nutné u tohoto typu synchronizace vyvinout mechanismy, které budou konflikty v synchronizovaných datech detekovat a řešit. Tyto konflikty a návrh jejich řešení jsou popsány v následující kapitole.



Obrázek 7: Lazy multi-master synchronization [27]

5.3 Problémy asynchronní multi-master synchronizace

Lazy multi-master synchronizace s sebou přináší několik možných problémů [22, 23]:

- Konflikty primárních klíčů.
- Konflikty během změny dat ve více databázích.
- Konflikty během změny a mazání dat ve více databázích.
- Duplicitní data.

5.3.1 Konflikt primárních klíčů

Při synchronizaci může dojít ke konfliktu během vkládání dat. Tento konflikt je způsoben konfliktem primárních klíčů, kdy dva nebo více uzlů sítě vytvořilo ve svých lokálních databázích záznamy, které se sice liší v datech ostatních sloupců, nicméně jejich primární klíče již existují v jiných databázích sítě.

Konflikty primárních klíčů vznikají při nevhodné volbě primárních klíčů. Nejčastěji pokud jsou v distribuovaném systému zvoleny jako klíče číselné hodnoty s konstantním přírůstkem [17].

Jednoduché číselné klíče s přírůstkem

Problém použití jednoduchých číselných klíčů s přírůstkem demonstruje tabulka 1.

Čas	Klient A	Klient B	Popis
6:00	Synchronizace dat	Synchronizace dat	Data konvergují
6:05	Vkládá záznam s ID 5	-	
6:10	-	Vkládá záznam s ID 5	
6:15	Synchronizace dat	Synchronizace dat	Konflikt primárních klíčů

Tabulka 1: Demonstrace vzniku konfliktů primárních klíčů v prostředí využívající jednoduché číselné klíče s přírůstkem

U číselných klíčů s přírůstkem dochází ke konfliktům v drtivé většině případů [17]. Konfliktům u tohoto typu klíčů se dá nicméně vyhnout použitím počáteční hodnoty klíče nebo volbou hodnoty přírůstku [6].

V prvním případě zvolíme pro každého klienta počáteční hodnotu klíče a předpokládáme, že počet záznamů v databázi nepřesáhne hodnotu vyhrazenou jinému klientovi. V druhém případě se zvolí přírůstek takový, aby došlo ke střídání hodnot klíčů (například lichá a sudá čísla) [6]. Je zřejmé, že ani jeden z navrhovaných způsobů nelze využít v případě klienta pro EEG/ERP Portál, neboť počet klientů není předem známý.

Další možností je využít znalosti topologie sítě, kdy Portál jako centrální uzel může záznamy vkládat s novými primárními klíči a ostatní uzly mohou vést informaci o tom, jakou hodnotu klíče přidělil Portál jejich záznamu. Takový způsob změny klíče vede ke značným ztrátám výkonu, protože se musí upravit i všechny cizí klíče, které určují relace k právě měněnému záznamu. Všechny tyto modifikace se musejí posílat zpět ke klientům, kteří musejí dodatečně upravit svoje databáze.

Přirozené klíče

Přirozený klíč je unikátní neprázdná hodnota, která v databázi jednoznačně určuje záznam tabulky a přirozeně se v ní vyskytuje. V ideálním případě je přirozený klíč neměnný. V případě změn je nutné neporušit referenční integritu.

V případě EEG/ERP Portálu lze například u tabulky PERSON použít jako primární klíč sloupec LOGIN, u kterého je zaručeno, že obsahuje jedinečné neprázdné hodnoty. Použití v jiných tabulkách je problematické, protože vede k nutnosti vytvářet kompozitní přirozený klíč z více sloupců.

Použití univerzálně jedinečného identifikátoru

Univerzálně jedinečný identifikátor (UUID) je 128 bitový řetězec zobrazovaný jako 32 hexadecimálních čísel s rozdělením na části pomocí pomlček [11]. V distribuovaných databázích se často využívá jako primární klíč, protože je obecně považován za jedinečný ve všech databázích sítě a existuje pouze malá pravděpodobnost, že by byl vygenerován duplicitní klíč [17].

Existuje několik různých verzí UUID, které se liší způsobem, kterým jsou vytvářeny [11]. Verze a variantu univerzálně jedinečného identifikátoru lze rozpoznat podle obsahu vyhrazených bitů. Strukturu UUID popisuje následující tabulka (tab. 2), ve které má jeden řádek velikost 32 bitů.

<table>
<tr><td colspan="3">Timestamp low</td></tr>
<tr><td colspan="2">Timestamp med</td><td>Timestamp hi + version</td></tr>
<tr><td>Clk_seq_hi + resrv</td><td>Clk_seq_low</td><td>Node (0-1)</td></tr>
<tr><td colspan="3">Node (2-5)</td></tr>
</table>

Tabulka 2: Struktura UUID verze 1 [11]

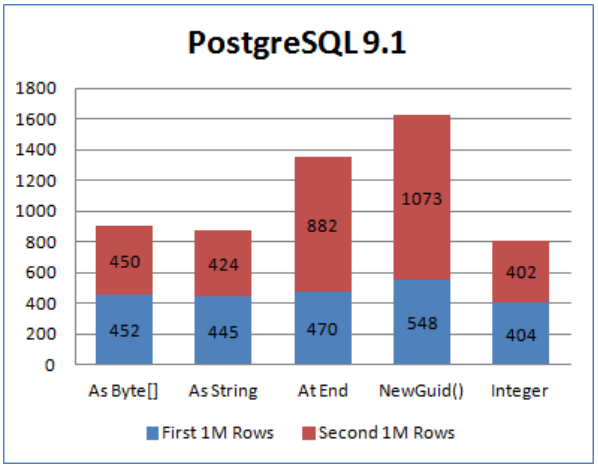
U verze 1 je univerzálně jedinečný klíč tvořen kombinací MAC adresy, která identifikuje zařízení, na kterém byl vytvořen, časového razítka a náhodné části. Tato kombinace poskytuje výhodu v tom, že je podle klíče možné určit zařízení, na kterém byl klíč vytvořen. Na druhou stranu však vede ke komplikacím v podobě používání více databází na jednom zařízení, a také případným podvrhem MAC adresy, což vede ke zvýšení rizika vygenerování konfliktního klíče. Druhá verze používá POSIX ID. Verze 3 a 5 je tvořena na základě zadaného řetězce buď hašovací funkcí MD5 nebo u páté verze algoritmem SHA-1 [11].

Čtvrtá verze unikátně jedinečného identifikátoru je tvořena náhodně [11]. Celkem je možné náhodně vygenerovat 2^{122} identifikátorů. Pravděpodobnost konfliktu u této verze je velmi malá, avšak rostoucí s počtem již existujících klíčů, lze ji vyjádřit vztahem (1):

$$p(n) = 1 - e^{-\frac{n^2}{2x}} \qquad (1)$$

n je počet vygenerovaných klíčů a *x* je celkový možný počet klíčů. UUID klíče nejsou tak kompaktní jako 32 bitové číselné klíče. To je jeden z důvodů, proč vede používání UUID ke zhoršení celkového výkonu. Tím dalším je jejich nesekvenčnost [26].

UUID lze generovat i sekvenčně, což například u PostgreSQL, kde je UUID datovým typem, vede ke snížení výkonu jen zhruba o 7,8 % oproti použití číselných klíčů (viz obr. 8). Ovšem s použitím mnoha klientů nelze sekvenčnosti dosáhnout, neboť jsou UUID generovány lokálně, to znamená, že lze dosáhnout nanejvýš lokální sekvenčnosti, která se na celkovém výkonu projeví jen minimálně.



Obrázek 8: Srovnání času (v sekundách) vkládání záznamů do PostgreSQL databáze se sekvenčními UUID reprezentovanými jako bytové pole nebo jako řetězec se sekvenční částí na začátku a na konci. Srovnáno s metodou GUID.NewGuid(), která generuje náhodný UUID, a přírůstkovými číselnými klíči. [26]

Kompozitní klíče s identifikátorem uzlu sítě

Při použití kompozitního klíče, který je tvořen z unikátního identifikátoru uzlu a jednoduché číselné hodnoty, nedochází k žádným rizikům konfliktů v primárních klíčích [17]. Další výhodou je znalost původu dat. Je možné dohledat, který uzel vložil jaký záznam. Použití kompozitních klíčů s identifikátorem uzlu však vede ke zvýšení doby prováděných dotazů (například u operací join). Kompozitní klíče také zesložiťují návrh aplikace využívající objektově relační mapování.

Protože je používání off-line klienta vázáno na konkrétní účet, lze jako část identifikátoru uzle použít e-mail, který je používán k přihlašování uživatele a je zajištěna jeho unikátnost. Nicméně vzhledem k tomu, že uživatel může používat více zařízení, došlo by ke konfliktům v primárních klíčích, je proto nutné přidat k identifikátoru i identifikaci zařízení. K tomuto účelu se hodí například použití MAC adresy.

5.3.2 Problém duplicitních dat

Při snaze vyhnout se konfliktům v primárních klíčích vzniká riziko vzniku duplicitních dat. Pokud některý z klientů vytvoří například nový záznam v tabulce HARDWARE a stejně tak je vytvořen záznam na jiném klientovi, mohou vzniknout duplicitní data (viz tab. 3).

ID	IS_DEFAULT	DESCRIPTION	TITLE	TYPE
1	1	Úsporný procesor	Intel 2100 T	Sandy Bridge
2	1	Úsporný procesor	Intel 2100 T	Sandy Bridge

Tabulka 3: Ukázka duplicitních záznamů

Z pohledu databáze se o duplicitní data nejedná, protože jsou jednoznačně určeny primárním klíčem, který se u obou záznamů liší, a žádný ze sloupců nemá nastavenou podmínku jedinečnosti. Protože se primární klíče liší, nedojde ke konfliktu během vkládání a data se vloží do databáze. Řešení duplicitních dat musí zajišťovat centrální uzel, který také zajistí, že při odstranění duplikátů budou upraveny i vazby objektů na tento duplikát.

5.3.3 Kolize během modifikace záznamů

V situaci, kdy více uzlů mění hodnoty stejného záznamu, dochází ke konfliktu během operace update (viz tab. 4). Během synchronizace je třeba vyřešit, která změna bude potlačena a která uplatněna. Detekce konfliktů se provádí několika způsoby. Jedním ze způsobů je použití časového razítka nebo verze záznamu. V takovém případě je nutné přidat příslušný sloupec do tabulek databáze. Další možností je k detekci konfliktu využít jen primární klíče nebo lze využít kombinaci primárního klíče a změněných dat.

Pokud je konflikt detekován, je třeba jej vyřešit. Existuje několik možných řešení. Stará data přepsat vždy novými nebo nová data ignorovat.

Čas	**Klient A**	**Klient B**	**Popis**
6:00	Synchronizace dat	Synchronizace dat	Data konvergují
6:05	Mění záznam s ID 2	-	
6:10	-	Mění záznam s ID 2	
6:15	Synchronizace dat	Synchronizace dat	Konflikt během updatu

Tabulka 4: Konflikt během změny záznamu

5.3.4 Kolize během mazání záznamů

Další konfliktní situací, která může nastat, je konflikt na operaci mazání. Dochází k němu tehdy, pokud jeden uzel záznam smazal a druhý změnil (viz tab. 5) [23].

Čas	**Klient A**	**Klient B**	**Popis**
6:00	Synchronizace dat	Synchronizace dat	Data konvergují
6:05	Maže záznam s ID 3	-	
6:10	-	Mění záznam s ID 3	
6:15	Synchronizace dat	Synchronizace dat	Konflikt během mazání

Tabulka 5: Konflikt během mazání a změny záznamu

Konflikt lze řešit tak, že se operace update nevykoná nebo se nahradí operací vkládání a do databáze se vloží nový záznam.

5.3.5 Další možnosti vzniku konfliktů

Během synchronizace může dojít ještě k dalším možnostem konfliktů, které je třeba řešit. Nicméně jejich řešení je jednodušší než řešení předchozích konfliktů. Jednou z možných situací je smazání téhož záznamu více uzly. V takovém případě se záznam smaže jen jednou a další operace mazání se nevykoná [15].

Dalším možným konfliktem je konflikt, kdy více uzlů vkládá stejný záznam. V takovém případě se záznam vloží jen jednou a další operací se místo vložení změní [15].

6 Řešení synchronizace klienta s Portálem

Změny datového modelu on-line Portálu s sebou přinášejí nutnost provést rozsáhlé změny v servisní vrstvě off-line klienta. Modifikace tříd, které zprostředkovávají synchronizaci, není triviální, protože je nutné dodržet integritu a konzistenci dat. Například je nutné vyřešit pořadí, v jakém se budou transakce provádět, vyřešit mechanismus opravy neproběhnutých transakcí a zajistit bezpečnost přenosu. Také je nutné řešit problémy konfliktů dat a jejich duplicit. Použitím hotového řešení se lze některým zmíněným komplikacím vyhnout.

6.1 Požadavky na synchronizační řešení

V předchozí kapitole jsem nastínil modely synchronizace a přístupy k ní. Celkem tedy vznikají čtyři možné kombinace [27]:

- Synchronní master-slave synchronizace
- Synchronní multi-master synchronizace
- Asynchronní master-slave synchronizace
- Asynchronní multi-master synchronizace

Pro potřeby off-line klienta lze využít jedinou možnost, kterou je asynchronní multi-master synchronizace, která s sebou přináší rizika v podobě konfliktů a také v podobě nekonzistentních dat. Synchronizační framework musí tento typ synchronizace zvládat a zároveň poskytnout nástroje, kterými lze řešit zmíněné problémy.

Klient a Portál používají odlišné databázové platformy, proto je vhodné, aby synchronizační framework podporoval synchronizaci heterogenních databází. Protože Portál a klienti operují s citlivými daty, je veliký důraz kladen na bezpečnost. Klient by tak neměl mít k dispozici přístupové údaje k centrální databázi.

Dalšími výhodami je možnost řízení synchronizace na straně on-line portálu, pozastavení synchronizace, výběr počátečních dat pro synchronizaci, filtrování synchronizovaných dat a jejich komprese.

6.2 Některá existující synchronizační řešení

6.2.1 Daffodil Replicator

Daffodil Replicator je nástroj určený pro synchronizaci, zálohu a migraci dat mezi různými databázovými servery. Daffodil Replicator je psaný v Javě, podporuje připojení přes JDBC ovladač a umožňuje práci s různými databázemi. K dispozici je open source varianta, která se liší oproti komerční variantě především ve výkonu. Daffodil Replicator umožňuje obousměrnou multi-master synchronizaci, detekuje a řeší kolize dat a umožňuje jejich filtrování [4].

Daffodil Replicator rozlišuje mezi dvěma typy uzlů: publisher a subscriber. Publisher vytváří publication, což je jedna nebo více tabulek určených k synchronizaci. Subscriber používá subscriptions k zjištění části dat určených k synchronizaci, které jsou určeny v publication odesílatele. Každý publisher musí mít jedinečnou IP adresu a číslo portu. Publisher může komunikovat zároveň pouze s jedním subscriberem. Synchronizaci dat inicializuje vždy jedině subscriber [4].

Nevýhodou je mimo nutnosti mít přidělenou jedinečnou IP adresu i ukončení vývoje produktu. Poslední verze open source varianty je 2.1 z roku 2006. Od té doby vývoj pokračoval pouze na komerční verzi (do roku 2011), kdy skončil i její vývoj. Není zajištěna jeho kompatibilita s novějšími verzemi Javy [4].

6.2.2 Sync4J

Snahou vývojářů Sync4J bylo navrhnout a implementovat framework, který by umožnil jakoukoliv synchronizaci různých databází, ke kterým existuje ovladač JDBC (Java Database Connectivity). Sync4J umožňuje autentifikaci uživatelů prostřednictvím JAAS (Java Authentication and Authorization Service). Sync4J se nyní nachází ve verzi 0.1 a jeho vývoj byl ukončen zhruba před třinácti lety, později byl přepracován k jinému účelu. Stránky projektu obsahují minimum dokumentace a z ní vyplývá, že Sync4J nepodporuje Apache Derby [7]. Z těchto důvodů se použití tohoto frameworku jeví jako nevhodné.

6.2.3 Rubyrep

Rubyrep je open source řešení pro asynchronní obousměrnou multi-master synchronizaci. Podporované databázové platformy jsou PostgreSQL a MySQL. Rubyrep nabízí automatické řazení operací ve snaze vyhnout se přidávání záznamů s vazbami na ty, které ještě nebyly vytvořeny. Konflikty dat jsou řešeny buď zachováním starých dat, nebo jejich přepsáním. Další způsoby mohou být dodatečně doprogramovány [12]. Rubyrep má zásadní nevýhodu v nutnosti přistupovat k databázi klienta, ale i Portálu, tím musí nutně obsahovat přístupové údaje k databázi Portálu, což představuje snadno zneužitelné bezpečnostní riziko.

6.2.4 DBReplicator

DBReplicator nabízí asynchronní obousměrnou multi-master synchronizaci mezi rozdílnými databázemi, včetně PostgreSQL a Apache Derby. Stejně jako Daffodil Replicator používá model Publisher - Subscriber. K tvorbě konfigurace je možné použít grafické rozhraní. DBReplicator umožňuje filtrování dat a základní, omezené řešení konfliktů dat. Synchronizaci je schopen provádět v určeném čase nebo nepřetržitě. DBReplicator byl uvolněn pod licencí GNU GPL2 [25].

6.2.5 Bucardo

Bucardo je synchronizační systém vytvořený v jazyce Perl a uvolněný pod licencí BSD. Umožňuje asynchronní synchronizaci mezi dvěma master uzly. Bucardo podporuje pouze databázovou platformu PostgreSQL a je možné jej používat pouze na unixových systémech [9]. Další informace, například o způsobu řešení konfliktů v datech, nejsou známy, protože existují pouze omezené informace na stránkách projektu.

6.2.6 SymmetricDS

SymmetricDS je software pro asynchronní synchronizaci databází v sítích s velkým počtem uzlů. Umožňuje filtrovat data určená k synchronizaci podle kritérií nebo je během procesu synchronizace pozměňovat. SymmetricDS komunikuje přímo jen s jednou databází a ostatními uzly sítě. Podporuje synchronizaci v libovolné topologii sítě a umožňuje uzly dělit do skupin, čímž usnadňuje návrh synchronizace [15].

SymmetricDS podporuje master-slave i multi-master synchronizaci. K dispozici je řada strategií, kterými lze detekovat a řešit konflikty v datech. SymmetricDS umí také provádět rollback nezdařených transakcí a díky pokročilému systému logování umožňuje absolutní kontrolu nad procesem synchronizace. Podporována je většina obvyklých relačních databází a také synchronizace souborů [15].

SymmetricDS existuje v open source verzi pod licencí LGPL nebo v profesionální komerční verzi, která se liší integrováním další komponent jako například pokročilého grafického administračního rozhraní a také poskytováním podpory ze strany společnosti JumpMind Inc., která SymmetricDS spravuje [10].

6.2.7 Srovnání vybraných synchronizačních řešení

Na základě získaných dat, které jsem prezentoval v předchozích kapitolách, jsem vytvořil srovnávací tabulku (tab. 6)[4, 7, 9, 10, 12, 15, 25].

X – funkce je obsažena O – omezená funkčnost ? – z dokumentace nelze zjistit	**SymmetricDS**	**Daffodil Replicator**	**Sync4J**	**Rubyrep**	**DBReplicator**	**Bucardo**
Multi-master, asynchronní	X	X	X	X	X	X
Detekce a řešení konfliktů	X	X	?	X	O	?
Podporované databáze[1]	X	X	O	O	X	O
Přímý přístup pouze k jedné DB	X	X	?	-	X	X
Podpora filtrování dat	X	X	-	X	X	X
Podpora kanálů[2]	X	-	-	-	-	-
Synchronizace souborů	X	-	-	-	-	-
Licence	GPL3	GPL2	?	MIT	GPL2	BSD
Rok posledního vydání	2014	2006	2001	2010	2008	2013

Tabulka 6: Srovnání vybraných synchronizačních řešení

Po srovnání všech synchronizačních řešení jsem zvolil SymmetricDS jako náhradu současného mechanismu synchronizace.

[1] Podporovanými databázemi se rozumí Apache Derby a PostgreSQL. Omezená podpora znamená podporu pouze jedné databázové platformy z výše uvedených.

[2] Podpora kanálů umožňuje rozdělit synchronizaci dat do více datových kanálů. Přenos dat kanálem pak probíhá nezávisle na ostatních kanálech.

7 SymmetricDS

Jako náhradu původního synchronizačního řešení jsem zvolil SymmetricDS. SymmetricDS umožňuje asynchronní mulit-master synchronizaci v sítích s mnoha uzly. Aplikace vznikala původně pro komerční účely, byla však uvolněna také jako open source software pod licencí General Public Licence (GPL 3.0) [15]. V současné době existují tedy dvě verze.

Komerční verze SymmetricDS PRO se od open source verze liší pokročilým grafickým administračním rozhraním, které ve volně šiřitelné verzi zcela chybí. Komerční verze také obsahuje instalátor, který umožňuje nastavení synchronizace krok po kroku. Posledním významným rozdílem je poskytnutí podpory ze strany společnosti JumpMind Inc., která SymmetricDS spravuje [15].

Open source verze SymmetricDS je v praxi často používaným řešením. Je nasazena například u projektů OpenMRS [24], což je software pro správu lékařských záznamů, a OpenBoxes[20], který se používá pro správu zboží a jeho distribuci.

7.1 Vlastnosti a funkce SymmetricDS

Následuje popis nejdůležitějších funkcí a vlastností SymmetricDS.

7.1.1 Obousměrná synchronizace

SymmetricDS umožňuje propagaci dat v obou směrech [15]. To znamená, že jej lze použít pro multi-master model synchronizace. Prováděné úpravy v lokální databázi off-line klienta se přenesou do databáze Portálu, stejně jako se přenesou změněná data v databázi Portálu do databáze klienta.

SymmetricDS umožňuje zvolit, zda má uzel propagovat změny v případě právě příchozích změn z jiných uzlů nebo zda má raději počkat na dokončení synchronizace dat a poté začít propagovat svoje změny. SymmetricDS umožňuje přijaté změny z jiného uzlu propagovat do ostatních uzlů s tím, že vynechá uzel, ze kterého změny obdržel, aby se tak vyhnul vzniku nekonečných smyček v synchronizaci (takzvaných synchronization loops) [15].

7.1.2 Řešení konfliktů dat

V předchozích kapitolách jsem popsal, jaké konflikty dat mohou nastat. SymmetricDS umožňuje zvolit, jakým způsobem jsou konflikty dat detekovány a jakým způsobem se mohou řešit. Strategie detekce a řešení konfliktů lze definovat zvlášť pro každé skupiny uzlů, zvlášť pro kanály anebo pro jednotlivé konkrétní tabulky databáze.

SymmetricDS pro detekci kolizí umožňuje použít tyto možnosti [15]:

- Použití pouze primárních klíčů.
- Použití starých hodnot před synchronizací.
- Použití nových příchozích hodnot.
- Použití časového razítka.
- Použití verze záznamu.

Poslední dvě možnosti vyžadují vytvořit v tabulce určené k synchronizaci sloupec s odpovídajícím datovým typem, který pak bude používán pro určení verze nebo času poslední změny záznamu.

Konflikty jsou detekovány během nahrávání příchozích dat do cílové databáze. V případě výskytu konfliktu lze zvolit tyto možnosti řešení konfliktů:

- Fallback

Fallback znamená, že se při selhání uzel stejně pokusí aplikovat změny, jako kdyby žádný konflikt nenastal. Jde tedy o snahu vždy aplikovat změny do databáze. Pokud byla konfliktní operace vkládání, mění se na update. Pokud se jednalo o update a záznam neexistoval, mění se operace na vkládání. Pokud se měl záznam smazat, ale už byl smazán, pak je operace ignorována.

- Ignorovat konflikt

Konflikt záznamu bude zcela ignorován. SymmetricDS umožňuje také ignorovat celou příchozí dávku.

- Vždy aktualizovat na novější data

Z konfliktních dat budou zachována ta data, která mají novější časové razítko nebo ta data, jejichž verze je vyšší.

- Manuální řešení

SymmetricDS umožňuje zaznamenávat chyby ve zvláštní tabulce. Tyto chyby pak mohou být řešeny ručně.

7.1.3 Detekce a oprava poškozených záznamů

Během odesílání dat mezi klientem a Portálem může dojít k poškození přenášených dat, například v případě náhlého výpadku internetového připojení v průběhu odesílání. V takovém případě se do databáze mohou uložit nekompletní data, která pak nejsou konzistentní.

SymmetricDS zaručuje, že záznamy budou vždy kompletní [15]. Veškeré chyby, které během synchronizace mohou nastat, jsou zaznamenávány. V případě chyby je záznam opětovaně odesílán, dokud nedojde k jeho úspěšnému přijetí cílovým uzlem. SymmetricDS umožňuje nastavit čas (timeout), po kterém jsou záznamy vymazány.

7.1.4 Filtrování dat a jejich transformace

SymmetricDS umožňuje filtrovat data, která mají být synchronizována. Umožňuje napsat vlastní podmínky pro výběr dat. V těchto podmínkách lze použít i jako proměnnou identifikátor uzlu. SymmetricDS umožňuje filtrovat data již při počátečním stahování dat.

Filtrování dat lze použít v situaci, kdy všichni uživatelé Portálu nemají mít k dispozici stejná data. Tato data pak mohou být ze synchronizace zcela vyjmuta. SymmetricDS umožňuje také vyjmout ze synchronizace určité sloupce. Toho lze využít například u synchronizace tabulky, která obsahuje hesla k uživatelským účtům.

SymmetricDS umožňuje také transformaci dat [15]. K dispozici je mnoho způsobů změny dat:

- Rozdělení jednoho sloupce na víc sloupců.
- Spojení více sloupců na jeden sloupec.
- Vložení konstantní hodnoty v závislosti na datech.
- Vložení řádků do cílové tabulky.
- Použít akci nadefinovanou ve vlastním BeanShell skriptu[3].

[3] BeanShell je skriptovací jazyk, který používá syntaxi jazyka Java [2].

7.1.5 Řízení synchronizace

SymmetriDS zaznamenává všechny dávky, které uzel odesílá nebo přijímá. Příchozí a odchozí dávky jsou zaznamenávány zvlášť. Je tedy možné vkládat dávky, které mají být odeslány nebo naopak mazat dávky, jejichž odesílání má být zrušeno. SymmetricDS také zaznamenává data, která jsou určena k synchronizaci. Tato data je opět možné modifikovat, přidat nebo odstranit.

SymmetricDS také umožňuje pozastavení synchronizace, její obnovení nebo vyřazení uzlu ze sítě. SymmetricDS lze řídit přiloženým nástrojem sys-admin, změnami jeho konfiguračních tabulek prostřednictvím SQL (Structured Query Language) dotazů nebo voláním metod jeho tříd. Je tak možné řídit celý proces synchronizace. SymmetricDS umožňuje automaticky propagovat změny provedené v jeho konfiguračních tabulkách ostatním uzlům v síti [15].

7.1.6 Synchronizace souborů

V novém návrhu databáze se předpokládá ukládání naměřených experimentů zvlášť mimo relační databázi do souborového systému. Takový přístup bude mít za následek zrychlení výkonu databázových dotazů nad relační databází, ale vyžaduje také vytvořit nový synchronizační mechanismus, který umožní přenášet velké soubory a zařazovat je v lokálním souborovém systému.

SymmetricDS umožňuje mimo relačních databází synchronizovat také soubory [15]. Je možné monitorovat změny v jednom nebo více adresářích. Soubory lze také synchronizovat ve všech podadresářích vybraných adresářů. Je možné použít zástupné znaky nebo regulární výrazy pro výběr názvů nebo typů souborů určených k synchronizaci. Soubory mohou být synchronizovány, když jsou vytvořeny, smazány anebo změněny. SymmstricDS také umožňuje nastavit, s jakou frekvencí jsou hlídány změny v souborech. Je také možné specifikovat skript, který se vykoná před nebo po zkopírování souboru. Synchronizace může probíhat obousměrně.

7.1.7 Datové kanály

SymmetricDS umožňuje rozdělit synchronizaci do datových kanálů. Synchronizace v jednotlivých datových kanálech probíhá nezávisle na ostatních. Synchronizace dat tak může probíhat paralelně, jen je třeba dodržet referenční integritu, například tak není možné rozdělit závislé tabulky do dvou kanálů. V případě chyby vzniklé v některém kanálu mohou ostatní kanály pokračovat v synchronizaci [15].

Lze také oddělit synchronizaci konfiguračních tabulek od vlastních dat. Také soubory je možné synchronizovat ve vlastním datovém kanálu a oddělit je od synchronizace databáze. Jedná se o defaultní nastavení.

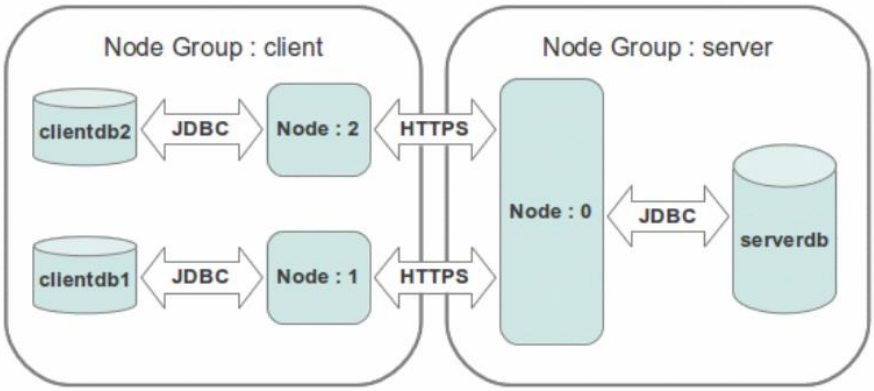
7.1.8 Jednoduchá implementace do on-line části

SymmetricDS může být nakonfigurován a posléze zabalen do webového archivu, který může být distribuován nezávisle na ostatních souborech on-line části. Webový archiv může být rozbalen na serveru jako webová aplikace souběžně s on-line Portálem. Implementace SymmetricDS pro on-line část nevyžaduje žádné její úpravy, neboť SymmetricDS pracuje přímo s databází serveru.

Je tak možné SymmetricDS kdykoliv pozastavit, ukončit nebo odstranit z on-line části aniž by byla narušena její funkcionalita. To poskytuje výhodu v případě nutnosti neprodleně odstavit synchronizační část z provozu. Všem off-line klientům je pak zabráněno v synchronizaci dat. Pokud je však SymmetricDS znovu spuštěn, může probíhající synchronizace pokračovat v momentě, ve kterém skončila. Je také možné deaktivovat možnost registrace nových uzlů do sítě, čímž je zabráněno přidávání nových klientů.

7.2 Průběh synchronizace

Po konfiguraci SymmetricDS, která je popsána v dalších kapitolách, může být zahájen proces synchronizace. SymmetricDS komunikuje přímo pouze s jednou databází přes JDBC (Java Database Connectivity Driver) a s dalšími uzly prostřednictvím HTTP (Hypertext Transfer Protocol) nebo HTTPS (Hypertext Transfer Protocol Secure). Příklad takové komunikace je zobrazen na obrázku 9.



Obrázek 9: Ukázka konfigurace sítě [15]

7.2.1 Přidání uzlu do sítě

Před samotným začátkem synchronizace je nutné provést tzv. registraci uzlu. K tomuto úkonu je třeba, aby se v síti vyskytoval uzel, který registrace umožňuje. Uzel, který žádá o registraci, má nastavenou registrační URL (Uniform Resource Locator), která ukazuje na umístění registračního uzlu [15]. V případě EEG/ERP portálu a klientů je registračním uzlem Portál.

Klient opakovaně žádá o registraci, přičemž v případě neúspěchu opakuje žádost za náhodný čas. V případě potřeby lze nastavit maximální počet neúspěšných registrací, ale to by bylo v tomto projektu nevhodné, protože by pak nikdy nedošlo k synchronizaci klienta s Portálem.

Registrační uzel lze nastavit tak, aby přijímal registrace automaticky. To znamená, že kdokoliv o registraci zažádá, bude registrován. Stačí mu pouze znát registrační URL. Nebo lze registrace provádět manuálně. V takovém případě se klient musí spojit s nějakou webovou službou Portálu a ta zařídí registraci po úspěšném ověření přihlašovacího jména a hesla. Registrace je provedena vložením záznamu obsahujícím identifikaci uzlu do příslušné konfigurační tabulky SymmetricDS [15].

Pokud byla registrace uzlu úspěšná a pokud je tak nastaveno, odešle Portál klientovi počáteční data. Tato počáteční data mohou být filtrována podle nastavených podmínek. SymmetricDS umožňuje, aby i klient, který o registraci žádá, po jejím úspěšném dokončení poslal svá počáteční data.

7.2.2 Konfigurace přidaného uzlu

Nastavení čerstvě přidaného uzlu do sítě je provedeno automaticky. Registrační uzel vybere data z některých svých konfiguračních tabulek a odešle je klientovi, který provede nastavení. Zároveň je provedena synchronizace databázových spouští (triggerů).

7.2.3 Výběr dat a jejich odeslání

SymmetricDS používá k určení změněných dat databázové spouště [15]. Databázové spouště jsou procedury, které jsou automaticky spuštěny jako reakce na specifickou akci v databázi [13]. SymmetricDS umožňuje nastavit tři typy spouští pro následující akce:

- INSERT
- UPDATE
- DELETE

SymmetricDS tak umožňuje zaznamenávat jen určité změny (například jen vložení nového záznamu). Ke každé spoušti lze nadefinovat podmínku, kdy je akce vykonána nebo se ignoruje.

Vybraná data jsou uložena do tabulky DATA. SymmetricDS posléze vytvoří dávku k odeslání. Dávka obsahuje identifikaci zdrojového a cílového uzlu, identifikátor kanálu, kterým se bude přenášet, čas vytvoření a změny dávky, stav dávky, její velikost a další informace sloužící například k určení případné chyby. Tyto informace jsou uloženy v tabulce OUTGOING_BATCH.

7.2.4 Příjem dat a jejich uložení

Po obdržení dávky jsou informace o ní uloženy do tabulky INCOMING_BATCH. Jedná se o stejné informace jako o ty, které jsou obsažené v tabulce OUTGOING_BATCH na zdrojovém uzlu. Následně jsou data extrahována a uložena do databáze. V případě, že se vyskytne nějaká chyba a data nelze do databáze vložit, je tato informace spolu s novými (ale i původními daty) uložena do tabulky INCOMING_ERROR, která také obsahuje identifikaci uzlu a identifikaci schématu a tabulky, v níž se chyba vyskytla [15].

8 Implementace

V této kapitole jsou popsány prostředky použité pro tvorbu klienta, struktura kódu klienta a serveru a provedené změny.

8.1 EEG/ERP Portál

Webové služby implementované na Portálu byly ponechány z důvodu zachování kompatibility s jinými projekty. Nicméně nyní je klient ke stahování a nahrávání dat nepoužívá. V celé on-line části došlo ke změně číselných klíčů na UUID. Vytváření těchto identifikátorů zajišťuje přímo Hibernate. Je toho dosaženo použitím následujících anotací.

```
@Id
@GeneratedValue(generator="system-uuid")
@GenericGenerator(name="system-uuid", strategy = "uuid2")
```

Kód 1: Použité anotace, které označují primární klíč a určují, jakým způsobem se má vytvářet. Strategie uuid2 generuje UUID kompatibilní se standardem RFC 4122.

SymmetricDS na straně Portálu běží samostatně na aplikačním serveru TomCat 7.0.53, což s sebou přináší výhodu nezávislosti na Portálu. Lze jej tak v případě potřeby kdykoliv ukončit a zároveň zachovat Portál v provozu.

Z funkcí popsaných v předchozí kapitole využívá tato implementace obousměrnou multi-master synchronizaci, detekci kolizí pomocí primárních klíčů a jejich řešení strategií fallback. Dále využívá filtrování dat u tabulky PERSON, kde jsou filtrována hesla. Synchronizovaná data jsou rozdělena do více kanálů. Poškozené záznamy jsou detekovány a opravovány automaticky. Synchronizace souborů není použita, neboť se soubory ukládají jako binární objekty přímo do databáze.

8.1.1 Struktura SymmetricDS on-line části

- **WEB-INF**
    - **classes –** obsahuje soubory s nastavením SymmetricDS
    - **lib** – obsahuje knihovny SymmetricDS
    - **web.xml** – deployment descriptor pro webovou aplikaci

Po rozbalení webového archivu se vytvoří logovací soubor, který je dostupný v adresáři $TOMCAT_HOME/logs/symmetric.log.

8.1.2 Konfigurace synchronizace

SymmetricDS běžící na Portále si jednak přečte svojí konfiguraci ze souboru symmetric.properties a jednak je nutné ho dodatečně nakonfigurovat. Dodatečná konfigurace je obsažena v přiloženém SQL souboru.

Po jeho vykonání dojde k vytvoření definice skupiny klientů a k nastavení jejich spojení pro obousměrnou synchronizaci. Zároveň jsou naplněny tabulky, podle kterých se vytvoří databázové spouště a datové kanály.

8.2 Off-line klient

8.2.1 Struktura off-line klienta

Program má následující strukturu

- **src/main/java** – obsahuje zdrojové kódy programu
- **src/main/resources**
    - **images** – obsahuje ikonu programu
    - **localization** – obsahuje soubory s texty, které program vypisuje
    - **wsdl** – obsahuje Web Services Description Language dokument
    - **log4j.xml** – obsahuje nastavení logování
    - **spring.xml** – obsahuje konfiguraci pro Spring
    - **elasticsearch-server.properties** – konfigurační soubor pro Elastic Search
- **pom.xml** – konfigurační soubor pro Maven
- **webservice.properties** – obsahuje adresu webové služby
- **symmetric.properties –** obsahuje nastavení SymmetricDS

Třídy programu jsou uspořádány do balíků.

- **cz.kiv.eegclient**
    - **data** – obsahuje datovou a servisní vrstvu
        - **dao** – obsahuje přístupové objekty pro záznamy uložené v databázi
        - **nosql** – obsahuje implementaci nerelační části
        - **pojo** – obsahuje objekty představující tabulky databáze
        - **service** – obsahuje servisní objekty

- **view** – obsahuje komponenty uživatelského rozhraní
  - **components** – obsahuje vlastní komponenty
  - **dialogs** – obsahuje dialogy programu
  - **tabs** – obsahuje balíky záložek hlavního okna programu
  - **MainFrame.java** – třída zobrazující hlavní okno programu
- **Main.java** – Třída určená ke spuštění programu

8.2.2 Použité prostředky aplikace

Všechny použité prostředky v dřívější verzi off-line klienta zůstaly zachovány. V některých případech došlo jen ke změně verze. Off-line klient nyní nově používá webový server Jetty 7.6.14, na kterém je instalován SymmetricDS 3.5.19.

8.2.3 Datový model

Datový model používaný klientem vychází z datového modelu používaného EEG/ERP Portálem, jen neobsahuje některé tabulky. Jedná se například o tabulky ARTICLES a ARTICLES_COMMENTS. Off-line klient je určen pro přidávání experimentálních dat, nepředpokládá se, že by byl využíván ke čtení článků. Oproti předchozímu datovému modelu chybí tabulka CHANGE a přibyly tabulky používané SymmetricDS. U existujících tabulek došlo ke změněně datového typu primárních klíčů. ERA diagram je kvůli rozměrům navrženého datového modelu uložen na přiloženém DVD.

8.2.4 Změny datové vrstvy

Se změnou datového modelu se změnila i datová vrstva klienta, ve které přibylo 36 tříd reprezentující nové tabulky datového modelu. K těmto třídám byly vytvořeny příslušné třídy DAO a jejich rozhraní. Jako vzory byly použity třídy používané Portálem.

8.2.5 Změny servisní vrstvy

Nejzásadnější změny byly provedeny v servisní vrstvě programu. Třídy zajišťující synchronizaci dat s Portálem byly odstraněny a nahrazeny třídou SyncDatabase, která zprostředkovává základní ovládání SymmetricDS.

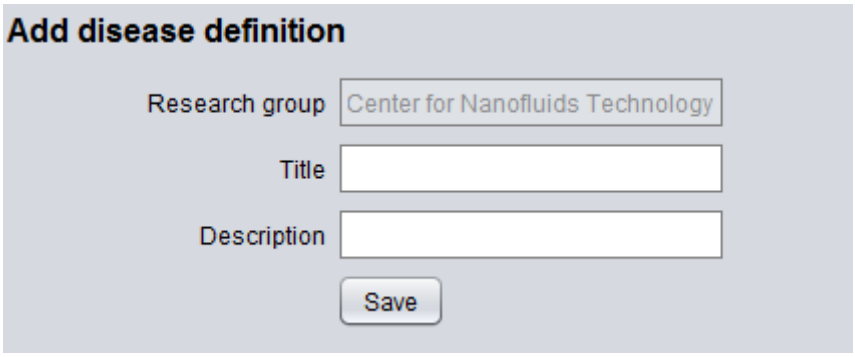
Konfigurace SymmetricDS probíhá jednak přečtením konfiguračního souboru symmetric.properties, čímž si klient vytvoří vlastní záznam o svojí identitě. Po registraci je další konfigurace zaslána Portálem a uložena do lokální databáze. Klient také obsahuje jednoduché metody pro spuštění SymmetricDS, jeho zastavení, zjištění, zda došlo k odeslání všech dávek, a metodu, která zjišťuje, zda byla přijata všechna počáteční data.

Servisní vrstva také obsahuje třídu, která spustí lokálně Elastic Search, aby jej mohl klient využívat pro ukládání dat do nerelační části. Tato třída obsahuje také metodu, která zjišťuje stav serveru. Nerelační část ovšem není v této verzi klienta podporována.

Zásadní změny byly také provedeny ve třídě DataServiceImpl (a jejím rozhraní DataService), která zprostředkovává vytvoření, výpis a kontrolu existence příslušných objektů.

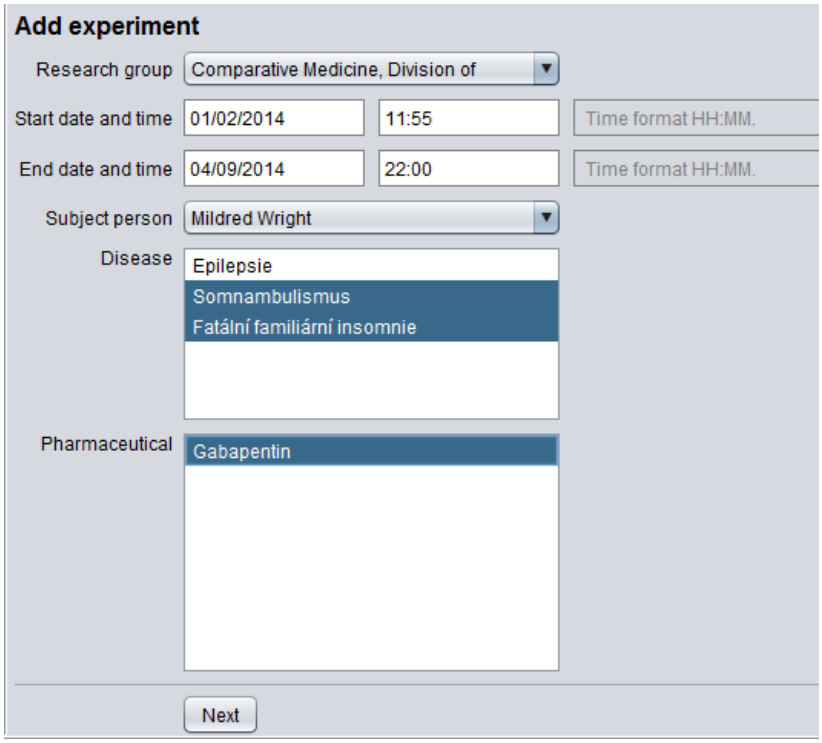
8.2.6 Změny prezentační vrstvy

V souvislosti s přidáním nových tabulek do databáze bylo nutné upravit prezentační vrstvu tak, aby bylo možné ukládat tyto nové objekty jako záznamy do databáze. (viz obr. 10)



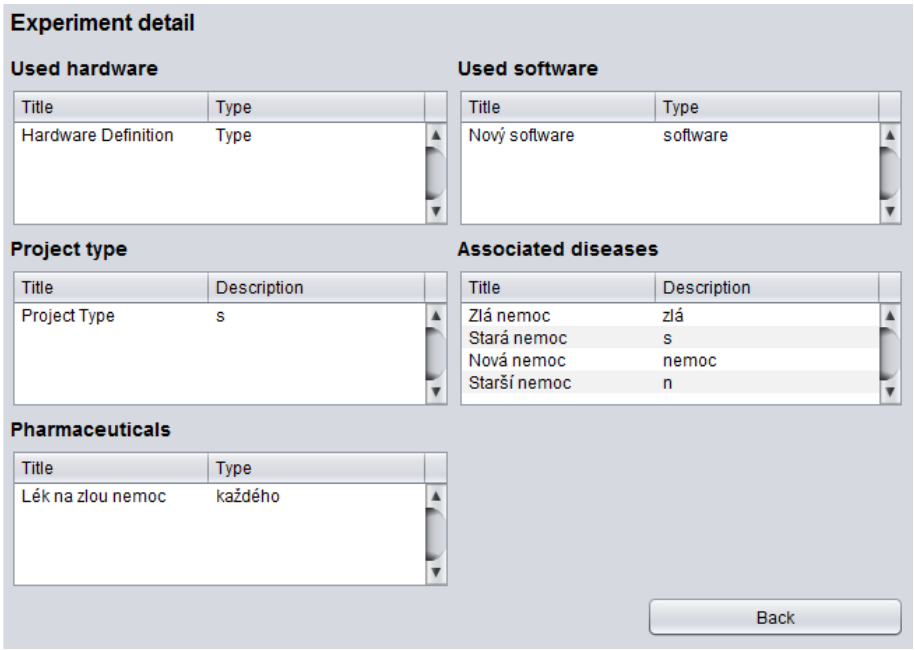
Obrázek 10: Ukázka průvodce pro přidání definice nového onemocnění

A také upravit průvodce pro přidávání nových objektů (viz obr. 11).



Obrázek 11: Ukázka průvodce pro přidání experimentu

Také bylo třeba upravit zobrazení existujících objektů a jejich detailů. (viz obr. 12)



Obrázek 12: Zobrazení pokročilého detailu experimentu

Za těmito účely bylo nutné přidat do prezentační vrstvy 26 nových tříd.

9 Testování

Aplikace byla podrobena dvěma funkčními testy a jedním zátěžovým testem. Funkční testy byly vykonány několika testery. Během testování došlo ke sběru připomínek testujících, na základě kterých došlo k úpravám aplikace.

Aplikace byla předložena k testování těmto testerům:

- Vysokoškolský student informatiky se znalostí návrhu aplikací.
- Vysokoškolský student biologie se základními počítačovými dovednostmi.
- Vysokoškolská studentka bankovnictví se základními znalostmi kancelářských programů.

9.1 Uživatelské testy

V dalších kapitolách jsou rozebrány jednotlivé testy, které byly předloženy těmto testerům.

9.1.1 Test svázání profilu a stažení počátečních dat

Předpokladem tohoto testu je funkční připojení k Portálu a funkční připojení k registračnímu serveru SymmetricDS.

- Zapněte aplikaci pomocí přiloženého souboru start.bat
- Zobrazí se seznam dostupných profilů
    - Stiskněte tlačítko New Profile
- Zobrazí se přihlašovací dialog – přihlaste se k účtu, který vám byl vytvořen
    - Zadejte login
    - Zadejte heslo
    - Stiskněte tlačítko Connect
- Zobrazí se zpráva o připojení k Portálu
    - Stiskněte tlačítko OK
- Zobrazí se výzva ke stažení počátečních dat
    - Stiskněte tlačítko Download
- Zobrazí se zpráva informující o úspěšném stažení dat
    - Stiskněte tlačítko OK

- Ověřte, zda došlo ke stažení všech dat
    - Klikněte na panel Experiments
    - Klikněte na panel Scenarios
    - Klikněte na panel Groups
    - Klikněte na panel People
    - Klikněte na panel Lists
        - Postupně projděte všechny číselníky

Výsledky

Všem testerům se podařilo úspěšně svázat profil a stáhnout počáteční data. Testeři neodhalili žádné anomálie v testovaných datech. Byla však odhalena chyba v původní aplikaci.

Pokud mají profily osob login se stejným prefixem, je vráceno více osob místo jedné. To způsobí chybu, která zabrání uživateli v přihlášení k aplikaci. Chyba byla opravena, chybující funkce byla nahrazena jinou.

9.1.2 Test obousměrné synchronizace

- Navštivte domovskou stránku Portálu
- Přihlaste se stejnými přihlašovacími údaji jako ke klientovi
- Klepněte na odkaz Experiments
- Vytvořte nový experiment klepnutím na Add experiments
    - Vytvořte nové definice entit
        - Nemoci
        - Softwaru
        - Hardwaru
        - Typu projektu
        - Léků
        - Počasí
        - Scénáře
        - Dalších parametrů osob a experimentů
- Dokončete přidávání experimentu
- Přejděte k off-line klientovi a přihlaste se
- Klepněte v horní liště na položku Sync
    - Zvolte možnost Connect
- Vyčkejte, dokud synchronizace neskončí

- Podívejte se, zda byl experiment přidán
    - Klikněte na Detail u příslušného experimentu
- Nyní vytvořte nový experiment v klientovi
- Klikněte na Add experiment v levém sloupci
- Pokračujte podle průvodce, vyberte požadovaná data a klikněte na Next
- Pokud pracujete v off-line režimu připojte se
    - Klikněte na Sync a zvolte Connect
    - V nabídce Sync a podnabídce Upload stats se podívejte, zda došlo k odeslání všech dávek dat
- Přihlaste se na Portál se stejnými uživatelskými údaji
- Klepněte na odkaz Experiments
- Klepněte na My Experiments
- Zkontrolujte, zda se data shodují

Výsledky

Všem testerům se podařilo data obousměrně synchronizovat. Bylo však odhaleno několik chyb, z nichž některé se vyskytovaly již v původní aplikaci.

Ve výpisu experimentů je chybně vypisováno datum experimentu. Místo měsíců se zobrazují dny (např. 30. 30. 2014). Chyba byla opravena, měsíce se nyní zobrazují správně.

Pokud jsou vybrané názvy entit příliš dlouhé, mělo by dojít k jejich zkrácení. Místo toho však dojde k tomu, že nejsou zobrazeny vůbec a nelze je vybrat. Chyba byla opravena, názvy se nyní zkracují, tak jak tomu bylo původně zamýšleno.

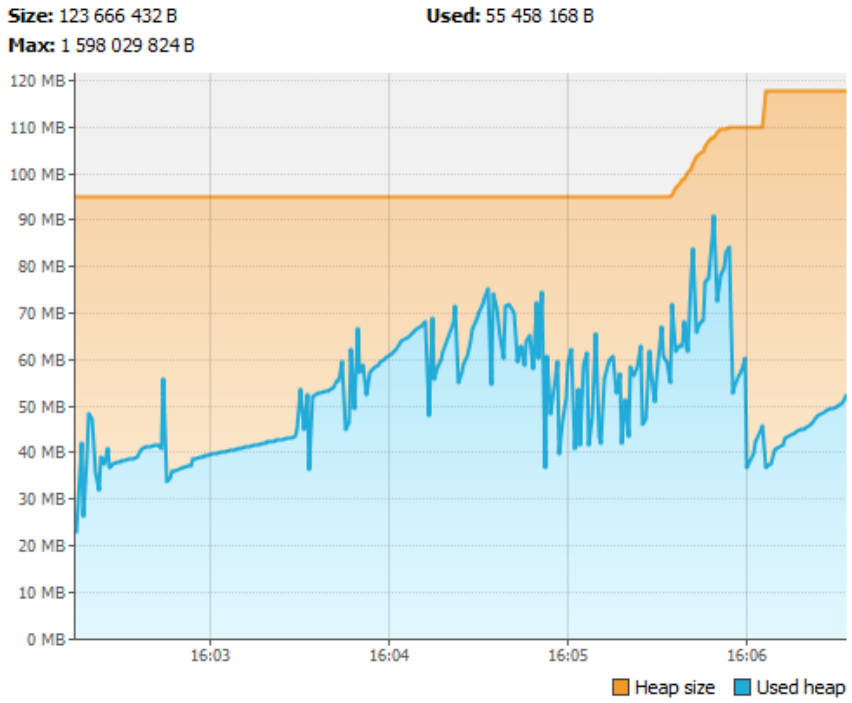
Změnou číselných identifikátorů na UUID dochází k tomu, že jako čísla experimentů nelze zobrazovat jejich identifikátory, protože se místo čísel zobrazí 36 znaků UUID. Tato neužitečná informace byla z výpisu smazána.

9.2 Zátěžový test

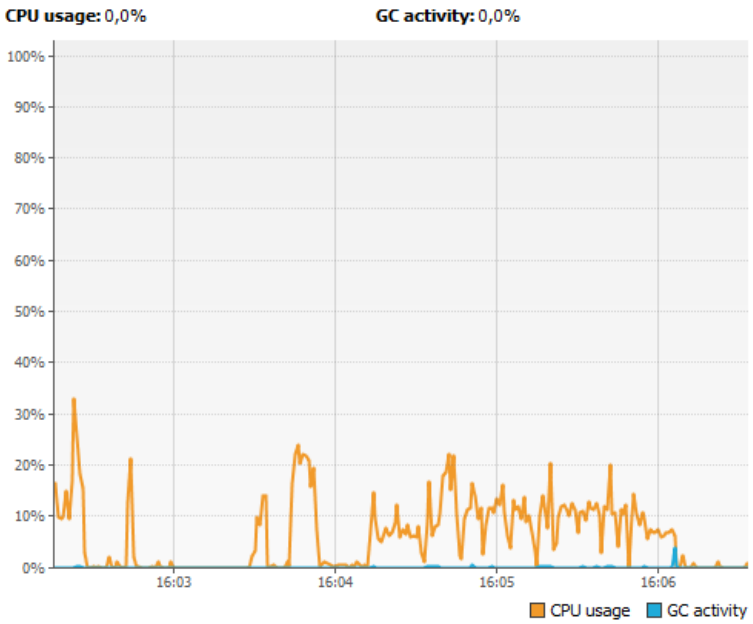
9.2.1 Test synchronizace tabulek s větším počtem záznamů

Cílem tohoto testu je zjistit, zda je synchronizace schopná synchronizovat databáze, které obsahují tabulky s větším počet záznamů. Předpokládaný počet záznamů v tabulkách je v řádech desetitisíců. K tomuto předpokladu byla zvolena odpovídající hodnota. Do tabulek HARDWARE a DISEASE bylo proto vloženo 100 000 řádků.

Výsledky testu ukázaly nevhodně nastavený kanál pro přenos dat. Byl vytvořen nový datový kanál s novým nastavením, který dokázal odeslat 200 000 záznamů klientovi ve velmi krátkém čase. Na následujících grafech je vidět využití paměti (obr. 13) a procesoru (obr. 14) klientem během počátečního stahování dat.



Obrázek 13: Využití paměti aplikací během synchronizace 200 000 záznamů v zátěžovém testu. Celková synchronizace trvala méně než 3 minuty.



Obrázek 14: Využití CPU aplikací během synchronizace 200 000 záznamů v zátěžovém testu.

10 Zhodnocení dosažených výsledků

V průběhu tvorby bakalářské práce jsem se seznamoval s velkým množstvím frameworků a technologií, které jsem dříve neznal. A protože se Portál vyvíjel v čase, začal jsem vytvářet prototyp aplikace pro jiný datový model a jinou databázovou platformu.

S přechodem na jiný datový model bylo třeba vyřešit nové problémy v synchronizaci off-line klienta. Seznámil jsem se proto s tím, jak se databáze v praxi synchronizují a jaké existují synchronizační přístupy a modely. Došel jsem k závěru, že nelze použít současnou verzi databáze Portálu, která používala číselné klíče, a zároveň zjistil, že synchronizace je příliš složitá a že existují nástroje, které mohou usnadnit její implementaci.

Předělal jsem aplikaci s využitím těchto nástrojů a upravil databázi Portálu. S těmito úpravami souvisely také nutné změny kódu Portálu a klienta. Výsledkem změn je funkční aplikace, kterou lze snadno rozšiřovat. Nový způsob synchronizace umožňuje obohatit aplikaci o funkce mazání a editaci záznamů.

Během testování aplikace byly odhaleny chyby. Některé se vyskytovaly již v původním off-line klientovi, některé byly nové. Nové chyby souvisely nejvíce s konfigurací synchronizace, kdy došlo k úpravě nastavení, čímž se synchronizace značně urychlila.

11 Závěr

Cílem této bakalářské práce bylo především upravit off-line klienta určeného k ukládání elektrofyziologických dat a metadat v době, kdy není dostupné internetové připojení. Úpravy zahrnovaly volbu nového datového modelu, úpravu datového modelu EEG/ERP Portálu, k němuž se klient připojuje, obou aplikací a synchronizace mezi nimi.

Abych mohl změny realizovat, musel jsem se seznámit nejprve s původní implementací off-line klienta, jeho datovým modelem a použitými prostředky. Stejně tak jsem se musel seznámit s několika různými verzemi Portálu, který se v průběhu času vyvíjel. Po analýze jsem navrhl datový model, který je inspirován datovým modelem používaným Portálem.

Prostudoval jsem teoretické principy synchronizačních modelů a přístupů a zjistil, které problémy se mohou během synchronizace vyskytnout. Analyzoval jsem řešení problému konfliktů primárních klíčů. Seznámil jsem se s řadou v současnosti používaných synchronizačních nástrojů a vybral z mého pohledu nejvhodnější variantu pro synchronizaci klienta s Portálem.

Naimplementoval jsem novou verzi off-line klienta a také upravil Portál. Zvolený synchronizační nástroj jsem nakonfiguroval a na straně klienta i integroval do aplikace. Aplikaci jsem důsledně otestoval a zjištěné chyby odstranil.

Budoucí vývoj off-line klienta a Portálu se může zaměřit na lepší integraci SymmetricDS do Portálu, na změnu datového typu primárních klíčů v PostgreSQL databázi, kterou nebylo možné dokončit z důvodu některých použitých metod frameworku Hibernate (findByParam). Díky použité synchronizaci bude také snadné rozšířit klienta o možnost editace a mazání záznamů.

Seznam zkratek

AOP – Aspect-orineted Programming

BSD – Berkeley Software Distribution

CXF – Apache Celtic XFire

DAO – Data Access Object

EEG – Electroencephalography

EP – Evokované potenciály

ER – Entity Relationship

ERP – Kognitivně evokované potenciály

GPL – General Public License

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

JDBC – Java Database Connectivity

JSON – JavaScript Object Notation

JSP – Java Server Page

JSTL – JavaServer Pages Standard Tag Library

LGPL – Lesser General Public License

MAC – Media Access Control

MIT – Massachusetts Institute of Technology

POJO – Plain Old Java Object

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

URL – Uniform Resource Locator

UUID – Universally Unique Identifier

XML – Extensible Markup Language

Seznam použité literatury a zdrojů informací

[1] BAREŠ, Martin. Kognitivní evokované potenciály. *Česká a slovenská neurologie a neurochirurgie: časopis českých a slovenských neurologů a neurochirurgů* [online]. Praha: Česká lékařská společnost J.E. Purkyně, 1993-, roč. 2011, č. 5 [cit. 2014-02-16]. Dostupné z: http://www.csnn.eu/ceska-slovenska-neurologie-clanek/kognitivni-evokovane-potencialy-36052

[2] BeanShell. *Introduction* [online]. 2014 [cit. 2014-04-21]. Dostupné z: http://www.beanshell.org/intro.html

[3] BJAALIE, J. G. a S. GRILLNER. Global Neuroinformatics: The International Neuroinformatics Coordinating Facility. *Journal of Neuroscience* [online]. 2007-04-04, vol. 27, issue 14, s. 3613-3615 [cit. 2014-02-16]. DOI: 10.1523/JNEUROSCI.0558-07.2007. Dostupné z: http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.0558-07.2007

[4] DAFFODIL SOFTWARE LIMITED. *Daffodil Replicator: Developer's Guide*. 2005. [cit. 2014-04-20] Dostupné z: opensource.replicator.daffodilsw.com

[5] DEL BENE, Andrea, Martin GRIGOROV, Carsten HUFE, Christian KROEMER, Daniel BARTL a Paul BOR. *Apache Wicket User Guide: Reference Documentation*. 2014. [cit. 2014-04-20] Dostupné z: http://wicket.apache.org/guide/guide/single.pdf

[6] FREEMAN, Robert a John GARMANY. [cit. 2014-04-20] *Oracle replication: Snapshot, Multi-master, Materialized Views Scripts*. 2nd ed. Kittrell, N.C: Rampant TechPress, 2003. ISBN 978-097-2751-339.

[7] FUNAMBOL. *Sync4J*. 2001. [cit. 2014-04-20] Dostupné z: http://sync4j.sourceforge.net/web/release/0.1/main.php?main=configuration.html

[8] CHOLAKIAN, Andrew. *Exploring Elasticsearch: A human-friendly tutorial for elasticsearch* [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://exploringelasticsearch.com/

[9] JENSEN, Jon a Greg MULLANE. *Bucardo*. [cit. 2014-04-20] Dostupné z: http://bucardo.org/wiki/Bucardo

[10] JUMPMIND INC. *SymmetricDS: Editions* [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://www.jumpmind.com/products/symmetricds/editions

[11] LEACH, MEALLING a SALZ. A Universally Unique IDentifier (UUID) URN Namespace. In: *RFC 4122* [online]. 2005 [cit. 2014-04-20]. Dostupné z: tools.ietf.org/html/rfc4122

[12] LEHMANN, Arndt. *Rubyrep*. 2009. [cit. 2014-04-20] Dostupné z: http://www.rubyrep.org/

[13] LISAL, Jan. *SQL - trigger, Databázové modelování: 6. přednáška z předmětu Datové struktury a databáze*. 2010. [cit. 2014-04-20] Dostupné z: http://www.nti.tul.cz/cz/images/0/03/DSD-6prednaska.pdf

[14] LIŠKA, František. *Off-line klient pro EEG/ERP portál*. Plzeň, 2013. Diplomová práce. Západočeská univerzita. Vedoucí práce Ing. Roman Mouček Ph. D. [cit. 2014-04-20]

[15] LONG, Eric, Chris HENSON a Greg WILMER. SymmetricDS User Guide: v3.5. [online]. 2013 [cit. 2014-05-03].
Dostupné z: http://www.symmetricds.org/doc/3.5/pdf/user-guide.pdf

[16] MICROSOFT. *.NET application architecture guide*. 2nd ed. Redmond, Wash.: Microsoft, 2009, xxxi, 524 p. Patterns. [cit. 2014-04-20] ISBN 07-356-2710-X.

[17] MICROSOFT. Selecting an Appropriate Primary Key for a Distributed Environment. *Synchronizing Databases* [online]. 2014 [cit. 2014-04-20]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb726011.aspx

[18] NOVOTNÝ, Jiří. *EEG/ERP portal security in new technologies*. Plzeň, 2012. Diplomová práce. Západočeská univerzita. [cit. 2014-04-21]

[19] NUNEZ, Paul L a Ramesh SRINIVASAN. *Electric fields of the brain: the neurophysics of EEG*. 2nd ed. New York: Oxford University Press, 2006, xvi, 611 p [cit. 2014-04-20]. ISBN 978-019-5050-387.

[20] OpenBoxes: Synchronization. [online]. [cit. 2014-05-04]. Dostupné z: https://code.google.com/p/openboxes/wiki/Synchronization

[21] ORACLE. Nimbus Look and Feel. In: *The Java Tutorials* 2014 [cit. 2014-04-17]. Dostupné z: http://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/nimbus.html

[22] ORACLE. *Oracle® Database Advanced Replication* [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://docs.oracle.com/cd/E11882_01/server.112/e10706/toc.htm

[23] PERIS, Ricardo Jiménez, Bettina KEMME, Marta PATIÑO-MARTÍNEZ. *Database replication*. San Rafael, Calif.: Morgan, 2010 [cit. 2014-04-20] ISBN 978-160-8453-818.

[24] SEATON, Mike. Database Synchronization with Symmetric DS. In: [online]. 2012 [cit. 2014-05-04]. Dostupné z: https://wiki.openmrs.org/display/projects/Database+Synchronization+with+Symmetric+DS

[25] TILMA, Brian. *DBReplicator User Manual*. 2008. [cit. 2014-04-20] Dostupné z: http://dbreplicator.org/DBReplicatorManual/index.html

[26] TODD, Jeremy. GUIDs as fast primary keys under multiple databases. [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://www.codeproject.com/Articles/388157/GUIDs-as-fast-primary-keys-under-multiple-database

[27] WIESMANN, PEDONE, SCHIPER, KEMME a ALONSO. Understanding Replication in Databases and Distributed Systems. *Understanding Replication in Databases and Distributed Systems* [online]. 2010 [cit. 2014-04-20]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.9166&rep=rep1&type=pdf

[28] ZÁPADOČESKÁ UNIVERZITA. *EEGBase* [online]. 2014 [cit. 2014-05-03]. Dostupné z: http://eegdatabase.kiv.zcu.cz

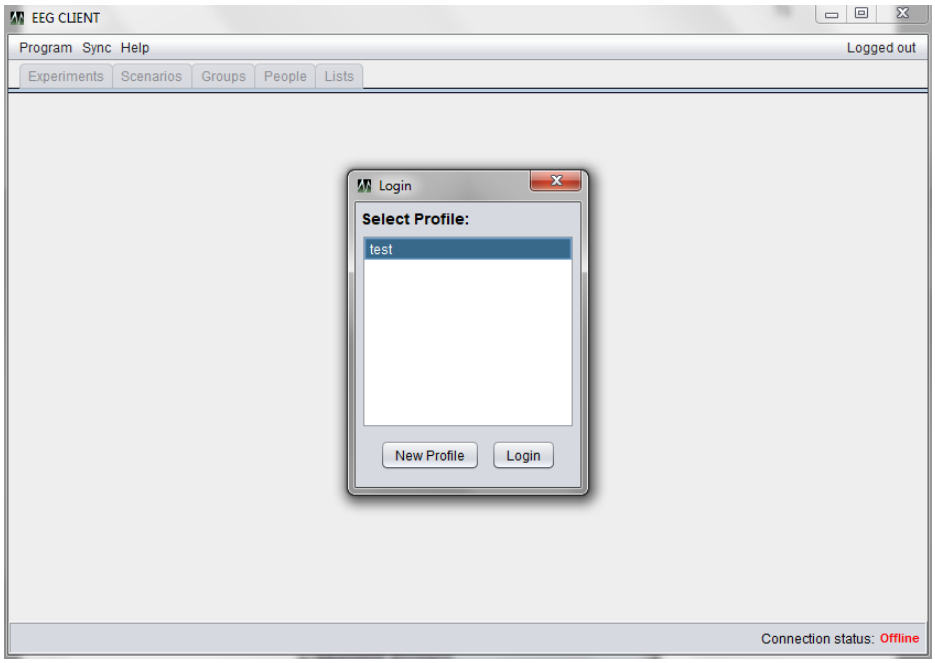
Seznam obrázků

Seznam tabulek

Přílohy

A. Uživatelská příručka

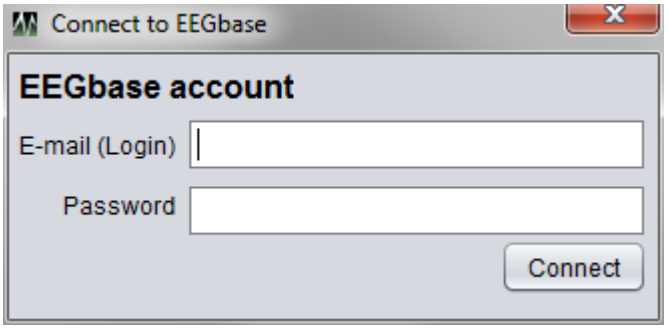
Po první spuštění programu se objeví nabídka (viz obr. A. 1), ve které si můžete zvolit existující účet nebo vytvořit nový.



Obrázek A. 1: Úvodní obrazovka po spuštění programu

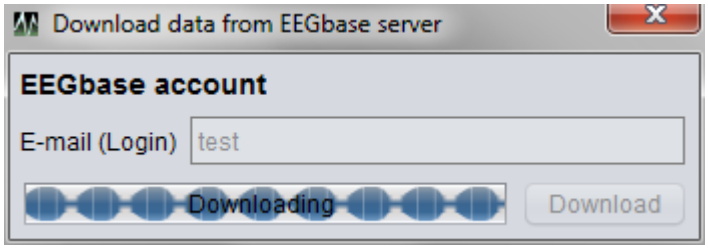
Pokud chcete vytvořit nový účet, klikněte na New Profile. Pokud se chcete přihlásit, vyberte profil ze seznamu a klikněte na Login.

Pokud vytváříte nový účet (viz obr. A. 2), zadejte jméno a heslo k účtu, kterým se přihlašujete na EEG/ERP Portál. Pokud se přihlašujete k existujícímu účtu, zadejte jen heslo.



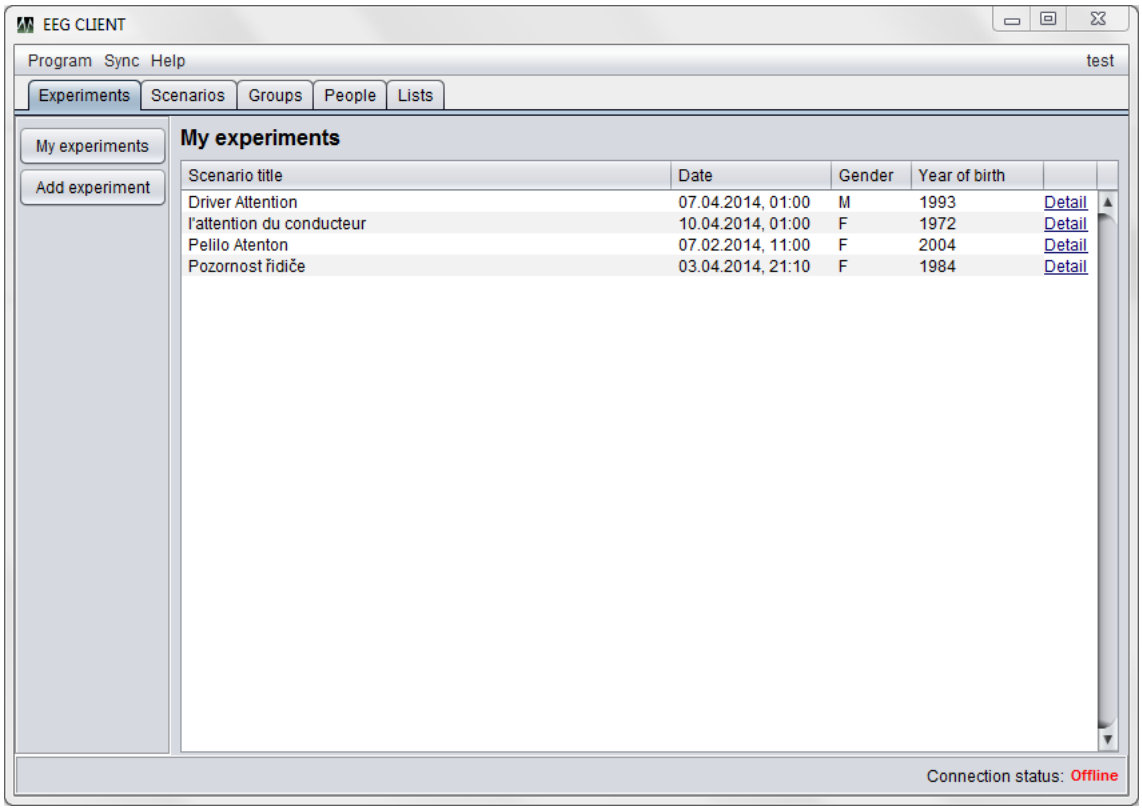
Obrázek A. 2: Vytvoření účtu

Po vytvoření účtu se zobrazí informace o tom, že jste připojeni k Portálu. Nyní je třeba stáhnout počáteční data, klikněte na tlačítko Download, čímž spustíte stahování dat (viz obr. A. 3).



Obrázek A. 3: Stahování dat

Po úspěšném stažení počátečních dat lze klienta využívat i v off-line režimu. Při příštím zapnutí klienta tak již stažení počátečních dat není nutné. Po potvrzení informace o stažení dat se zobrazilo hlavní okno programu (obr A. 4).



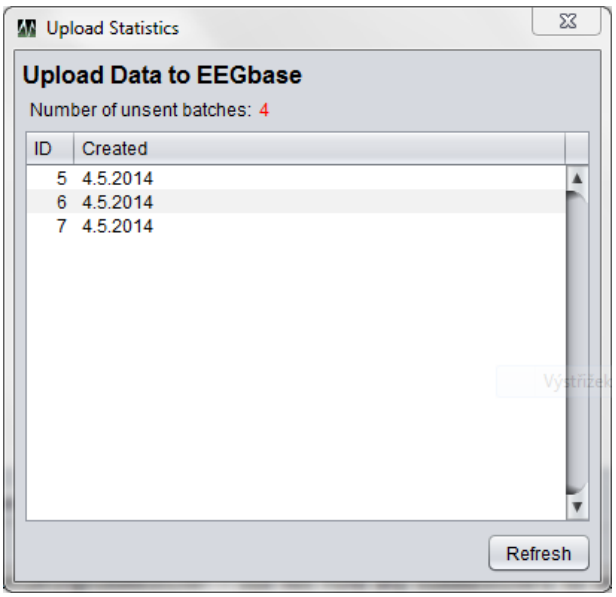
Obrázek A. 4: Hlavní okno aplikace

V pravém horním rohu vidíte aktuálně přihlášeného uživatele, pokud na něj kliknete, můžete se pomocí tlačítka odhlásit.

Dále vidíte horní menu s následující nabídkou:

- Program
    - Exit – ukončí program
- Sync
    - Connect – připojí se k Portálu
    - Donwload – ověří, zda jsou počáteční data stažena korektně
    - Upload Changes – zobrazí okno informující o změnách, které je třeba nahrát na server
    - Disconnect – odpojí se od serveru
- Help
    - About – zobrazí informace o programu

Pokud kliknete na Sync – Upload Changes, otevře se okno s informacemi o dávkách (obr. A. 5), které je třeba odeslat, než bude databáze synchronizována s databází Portálu. Mějte na paměti, že dávky se vytvářejí, pouze pokud jste připojeni. Jinak tato informace není k dispozici.



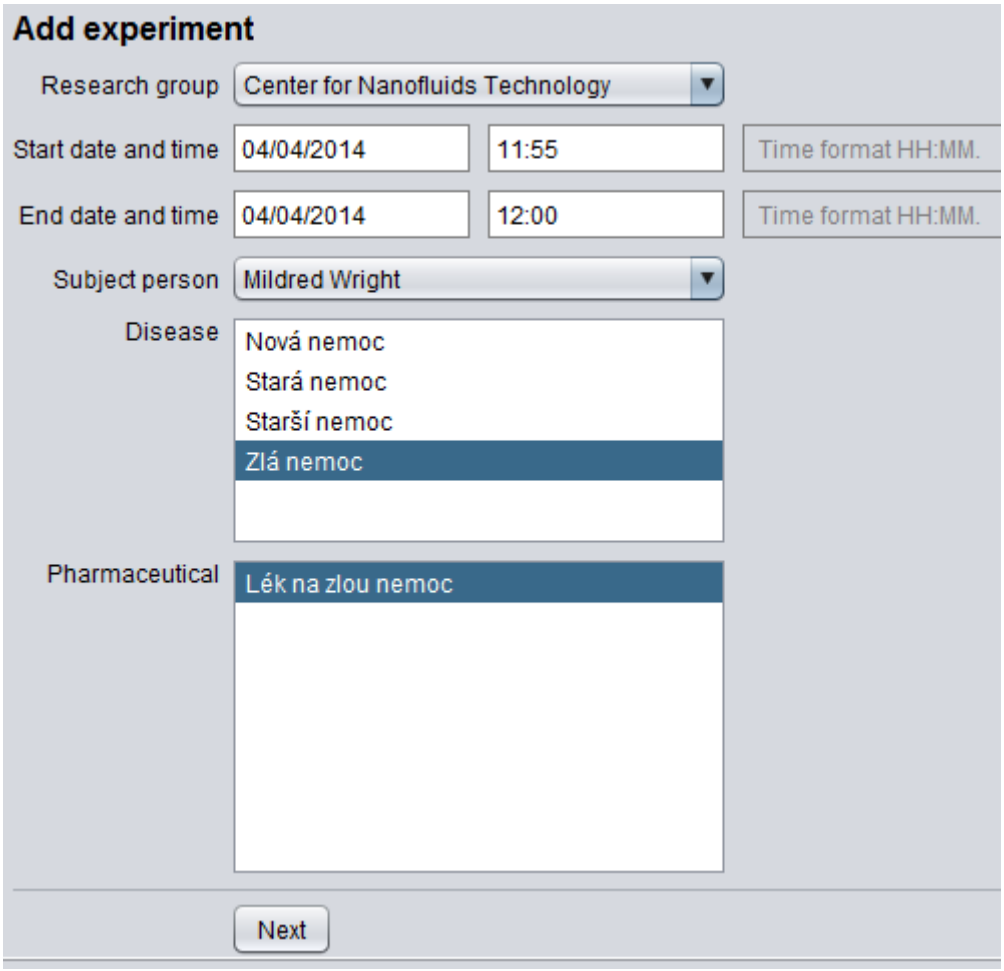
Obrázek A. 5: Seznam dávek, které je třeba odeslat

Zobrazují se jen dávky obsahující data. Celkový počet všech dávek (tedy i těch, které nesou konfiguraci) je uveden v horní části okna. Kliknutím na Refresh obnovíte statistiku.

Pod horní nabídkou jsou vidět záložky, kliknutím na ně zobrazíte informace o požadovaných objektech. Levý sloupec je pak podnabídkou aktuálně otevřeného panelu.

Přidávání entity

Klikněte na Add experiment k přidání nového experimentu. V novém okně vyberte, jak má experiment vypadat.

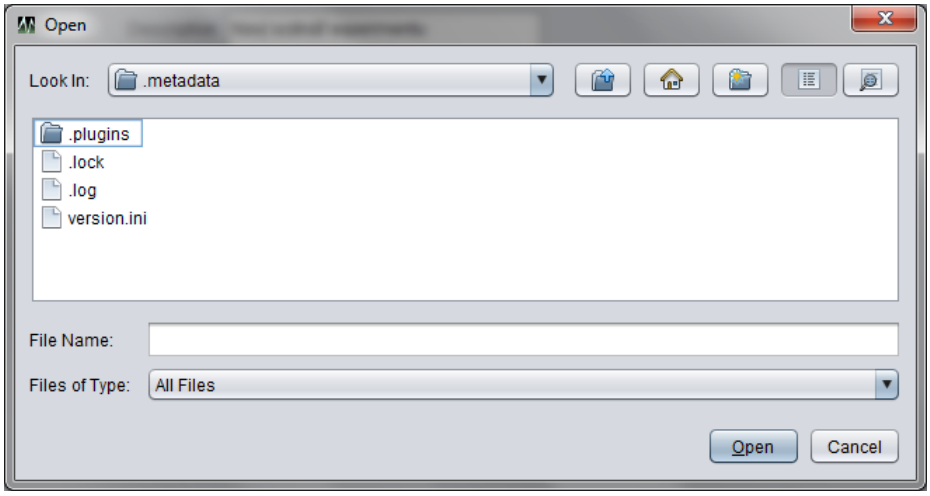


Obrázek A. 6: První okno přidání experimentu

Všichni průvodci (obr. A. 6) mají jednotnou formu a stejné prvky. Výběr z rolovacího okna provedete kliknutím na něj, rolováním se posunete na požadovanou pozici a kliknutím na objekt jej vyberete.

Kliknutí do polí typu datum se otevře kalendář, ve kterém zvolte příslušné datum. V prvku typu list vyberte položku nebo více položek. Při výběru více položek držte CTRL pro obarvování položek.

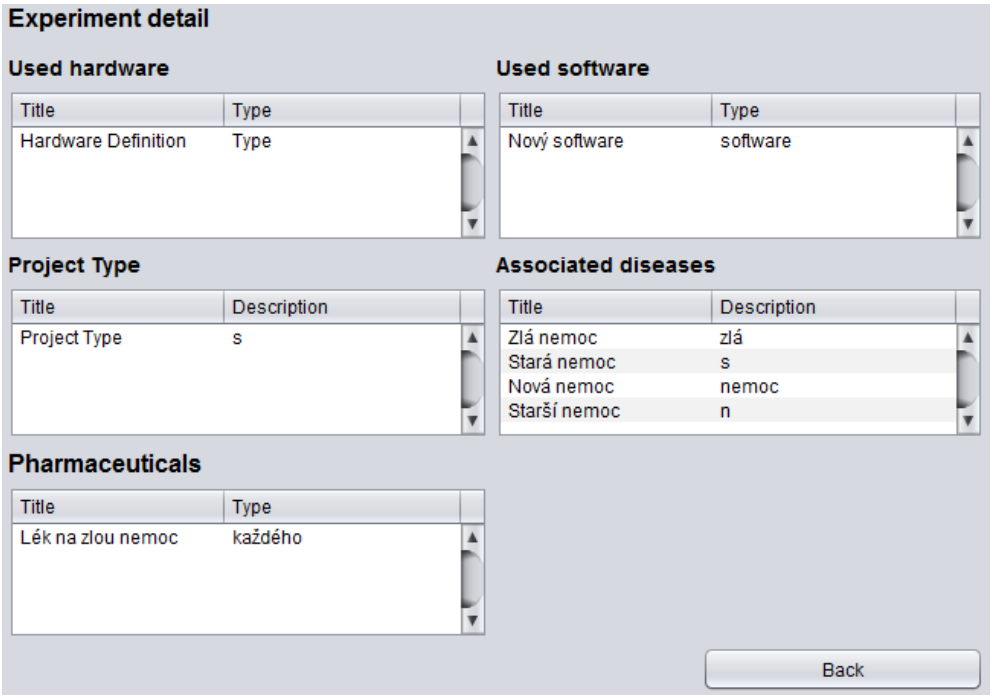
Pokud je třeba vybrat soubor, otevře se okno pro výběr souboru (viz obr. A. 7).



Obrázek A. 7: Výběr souboru

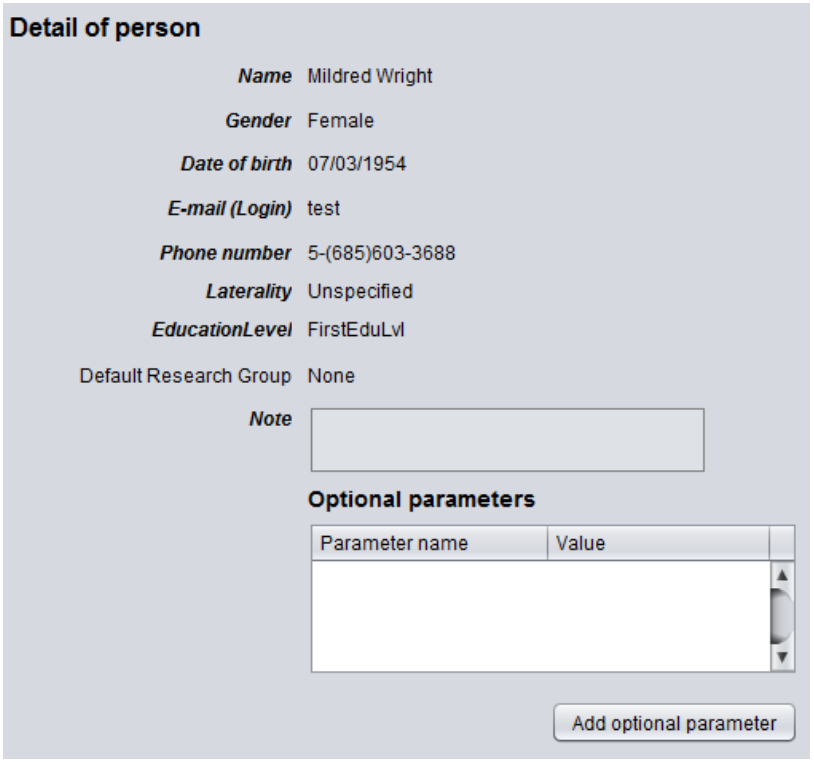
V takovém případě zvolte soubor a stiskněte Open. Posledním možným prvkem je zaškrtávací pole. Kliknutím na něj jej zaškrtnete a dáváte najevo, že uvedená vlastnost objektu platí. Dalším kliknutím můžete zaškrtnutí zrušit. Pro návrat používejte tlačítko Previous, pro pokračování tlačítko Next. Pokud chcete objekt uložit, klikněte na Save.

Pokud chcete zobrazit detail objektu, klikněte na tlačítko detail, které se vedle něj nachází. Nezapomeňte, že další podrobnosti experimentu můžete zobrazit kliknutím na Show Detail na kartě s detaily experimentu (obr. A. 8).



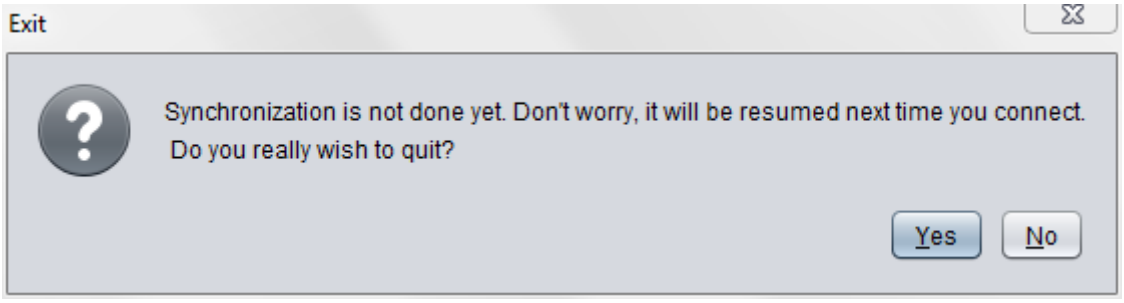
Obrázek A. 8: Pokročilé detaily experimentu

Parametr k objektu přidáte tlačítkem Add optional parameter u příslušné tabulky parametrů. Na obrázku vidíte tabulku parametrů u tabulky osob.



Obrázek A. 9: Detail osoby

Pokud zahájit synchronizaci dat, stačí kliknout na Sync – Connect a zadat heslo. Synchronizace začne automaticky. Během synchronizování můžete pokračovat v práci. Změny se budou projevovat automaticky, jakmile to bude možné. Pokud budete chtít ukončit program, ale ten bude obsahovat ještě neodeslané dávky, budete na tuto skutečnost upozorněni (viz obr. A. 10).



Obrázek A. 10: Potvrzení ukončení programu

Kliknutím na No se můžete vrátit zpět do programu.