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

Bakalářská práce

Návrh a implementace mobilní aplikace pro sběr vybraných medicínských dat

Plzeň 2016 Jiří Matyáš

Místo této strany bude zadání/oboustranná kopie zadání

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

V Plzni dne 2.5.2016

……

Jiří Matyáš

Abstract

Mobile devices, or applications installed on them, are currently able to monitor user's health status information. This information is used only for this application's purposes and is not stored in any medical documentation. The main goal of this bachelor thesis is implementation of mobile application that will collect sleep medical data and store them in personal medical documentation which is being developed at the Department of computer science and engineering of University of West Bohemia in Pilsen. OpenEHR archetype will formally describe this data. Application will collect data from user as well as from third-party applications. Sleep terminology will be submitted certified ontologies. For storing unstructured medical data, is necessary to explore appropriateness of use non-relational databases.

Abstrakt

Mobilní zařízení, respektive aplikace na nich nainstalované, jsou v současné době schopny monitorovat různé zdravotní informace o stavu jejich uživatele. Tyto informace však využívají pouze pro své účely a neukládají je do žádné zdravotní dokumentace. Cílem této práce je implementování mobilní aplikace, která bude sbírat spánková medicínská data v rámci osobní zdravotní dokumentace vyvíjené na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni. Tato data bude formálně popisovat openEHR archetyp. Aplikace bude data získávat od uživatele a rovněž z aplikace třetí strany. Spánková terminologie bude podložena ověřenou ontologií. Pro ukládání nestrukturovaných medicínských dat je nutno prozkoumat vhodnost použití nerelačních databází.

Poděkování

Rád bych poděkoval Ing. Václavu Papežovi za jeho trpělivost, cenné rady a kvalitní vedení průběhu práce. Dále bych chtěl poděkovat svým blízkým a přátelům za jejich podporu.

OBSAH

1. ÚVOD

V současnosti je stále větší nutností používat informační a komunikační technologie pro zefektivnění pracovních postupů a snadnější práce s informacemi. Elektronické zdravotnictví se zaměřuje na použití těchto technologií v medicínském prostředí. V elektronickém zdravotnictví je kladen důraz zejména na bezpečnou práci s daty, efektivní spolupráce zdravotnických systémů a poskytování kvalitní zdravotní péče lidem. Pro řešení těchto úkolů existuje několik přístupů, které nabízí například mezinárodně uznávaná organizace openEHR. Tato nezisková organizace vyvíjí referenční model pro vhodnou implementaci elektronické zdravotní dokumentace.

Elektronická zdravotní dokumentace je důležitou součástí elektronického zdravotnictví, do které jsou ukládány zdravotní záznamy pacientů. Tyto záznamy by měly být použitelné i mimo zdravotnické zařízení, které je vytvořilo. Například krevní obraz pacienta vytvořený v nemocnici může využívat pacientův praktický lékař, který nemusí obraz provádět znovu. To zvyšuje efektivitu poskytování zdravotní péče. Pro pacienta by měla být zdravotní dokumentace přístupná k nahlédnutí na internetu. Pacient je tím informovaný o svém zdravotním stavu.

Pacient dále může sledovat svůj zdravotní stav pomocí vytváření vlastních zdravotnických záznamů (např. měření krevního tlaku), které může vkládat do své zdravotní dokumentace. Mobilní zařízení, respektive vybrané aplikace na těchto zařízení, mohou některé zdravotní informace o stavu jejich uživatele zaznamenat. Zatím je však neukládají v rámci žádné elektronické zdravotní dokumentace. Osobní elektronická zdravotní dokumentace zatím v České republice neexistuje, a to i přes nadějné pokusy.

Cílem této bakalářské práce je vytvořit mobilní aplikaci, která bude schopná sbírat medicínská data s využitím konceptu openEHR. Data bude formálně popisovat vhodný openEHR archetyp a zdroj těchto dat bude aplikace třetí strany a uživatel aplikace. Jako příklad sbíraných dat v rámci této práce byla zvolena spánková data. Vhodný archetyp pro popis spánkových dat zatím neexistuje. Proto je nutné ho nejprve vytvořit. Dále je nutné zvolit vhodné úložiště dat na mobilním zařízení, které bude schopné data synchronizovat se vzdáleným úložištěm dat. Vzdálené úložiště bude navázané na osobní zdravotní dokumentace, která je v současné době vyvíjena skupinou Medical Informatics na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni.

2. ELEKTRONICKÉ ZDRAVOTNICTVÍ

Elektronické zdravotnictví neboli eHealth zahrnuje veškeré implementace informačních a komunikačních technologií, které souvisí se zdravím nebo medicínou. Tyto aplikace slouží pro poskytování zdravotní péče lidem ve všech oblastech zdravotnictví. [1]

Hlavní účel eHealth nástrojů spočívá v používání zdravotnickými pracovníky, pro administraci pacientů a jejich zdravotních záznamů, podporu rozhodování v péči o pacienta, jako například určení správné medikace. Zdravotnická zařízení mohou dále používat nástroje elektronického zdravotnictví pro plánování logistiky, zpracování laboratorních výsledků, elektronickou komunikaci na úrovni zdravotnických pracovníků i na úrovni zdravotnických zařízení, telemedicínu atd. Nástroje elektronického zdravotnictví mohou sloužit rovněž pro výukové účely. Také je může používat pacient při hledání medicínských informací online. Základní stavební kameny elektronického zdravotnictví jsou elektronické zdravotní záznamy EHR (Electronic Health Record). [1]

2.1 EHR systémy

EHR systémy jsou postavené na sběru medicínských dat v elektronické podobě. Tato data nemusí poskytovat pouze lékař nebo zdravotnické zařízení, ale jakákoliv entita dodržující určitá pravidla pro medicínská data. Systémy mohou dále poskytovat nástroje pro zpracování těchto dat, které může například lékař využít při rozhodování o péči pacienta. Mnoho rutinních medicínských postupů lze používáním těchto nástrojů automatizovat a zefektivnit. [2]

2.1.1 Elektronické zdravotní záznamy

EHR označuje elektronický záznam z digitální zdravotní dokumentace pacienta. EHR záznamy jsou vytvářeny v reálném čase v různých zdravotnických zařízeních a mohou být centralizovány pro jejich pozdější použití. Autorizovaným uživatelům je následně umožněn bezpečný přístup k centralizovaným EHR záznamům. Více lékařů ošetřující jednoho pacienta může v případě potřeby a se souhlasem pacienta jednoduše na svém počítači přistupovat k záznamům z různých vyšetření pacienta. To by mělo velice usnadnit a zefektivnit práci zdravotnickým pracovníkům a tím zkvalitnit zdravotnickou péči o pacienta. EHR záznam

může obsahovat zdravotní historii, diagnózy, medikace, radiologické obrázky, alergie, laboratorní výsledky atd. [2]

Podobné EHR záznamům jsou záznamy EMR (Eletronic Medical Record). Tyto záznamy jsou založené na podobném principu jako záznamy EHR, ale jsou shromažďovány a využívány v rámci jednoho zdravotnického zařízení. Začaly být používané jakožto první digitální podoba papírové zdravotní dokumentace pacienta ve zdravotnickém zařízení. Nevýhodou je, že tyto záznamy nejsou použitelné vně daného zdravotnického zařízení, na rozdíl od EHR záznamů. [3]

Dalším druhem elektronických zdravotních záznamů jsou záznamy PHR (Personal Health Record). Tyto záznamy vytvářejí a udržují sami pacienti a slouží pro udržování informací o jejich zdravotním stavu. PHR záznamy mohou pocházet z různých zdrojů, např. od poskytovatelů zdravotní péče, ale i pacientů samotných. PHR pomáhají pacientům sledovat svůj zdravotní stav (dietní plán, sledování hladiny cukru v krvi, plán užívání medikací atd.). PHR jsou striktně odděleny od oficiálních elektronických záznamů zdravotnických zařízení a nesmí nahrazovat EHR. Mohou však být velice nápomocné lékařům při stanovení diagnózy a při komunikaci pacienta s lékařem a naopak. [4]

2.1.2 EHR v EU

Dnes již v mnoha zdravotnických zařízeních v zemích Evropské unie informační a komunikační technologie a další eHealh nástroje úspěšně fungují. Bohužel jsou implementace těchto systémů velice rozdílné a nejsou vůči sobě příliš interoperabilní. Ve většině případů volila zdravotnická zařízení nejlepší řešení pro sebe. To brání úspěšné komunikaci napříč všemi zdravotnickými systémy v Evropské unii. [5]

Pro Evropskou unii existuje takzvaný Akční plán eHealth. Tento plán stanovuje kroky pro elektronizaci zdravotnictví v rámci EU. Jedná se o konkrétní akce pro elektronizaci zdravotnictví v rámci Evropské unie. Akční plán byl vypracovaný v roce 2011. Měl by upevňovat již provedené kroky pro elektronizaci zdravotnictví a dále vytvářet nové. Další cíl plánu je podpora členských států v této oblasti v nejlepším zájmu pacientů. Plán dále podporuje inovace ve zdravotnictví. [5]

2.1.3 EHR u nás

V České republice se o elektronizaci zdravotnictví stará Národní plán rozvoje eHealth. Tento plán určuje instituce České národní fórum pro eHealth a ICT unie ve spolupráci s Ministerstvem zdravotnictví. Cílem tohoto plánu je vytvořit koncept elektronického zdravotnictví, který by byl politicky přijatelný a zajišťoval by vysokou kvalitu a dostupnost zdravotní péče. [6]

Koncept by měl zahrnovat legislativní řešení a standardizaci elektronického zdravotnictví. Dalším bodem je vytvoření fungující elektronické zdravotní dokumentace, kterou by používali pacienti i zdravotničtí pracovníci. Ke správnému používání této dokumentace je nutné vytvořit elektronickou identifikaci pacientů a zdravotnických pracovníků. V neposlední řadě je nutné vytvořit program pro vzdělávání zdravotnických pracovníků a pacientů v používání dokumentace. [6]

Velkým rizikem realizace jakéhokoliv projektu elektronického zdravotnictví v České republice je nízká motivace lékařů, pacientů a státních institucí. Dalším rizikem je nedostatečná legislativa, která není na elektronické zdravotnictví připravena. V případě úpravy pro elektronické zdravotnictví by se musel měnit zákon 20/1966 Sb. o zdraví lidu, což by vedlo k dlouhému legislativnímu procesu zatíženého politickou situací. Posledním velkým rizikem je financování těchto projektů. Ve zdravotnictví a zdravotnických zařízení je špatná finanční situace, která také není připravená na takové změny. [6]

Největším projektem elektronické zdravotní dokumentace byl systém elektronických zdravotních knížek IZIP, který byl v roce 2004 uveden do provozu. Tento projekt vytvořila společnost IZIP a.s. s finanční podporou majoritního vlastníka této společnosti Všeobecnou zdravotní pojišťovnou. Elektronická zdravotní knížka představovala souhrn zdravotních informací pacienta a měla sdílet informace o pacientech mezi lékaři, kteří mezi sebou sdíleli výsledky různých vyšetření. Dále měla poskytovat informace samotnému pacientovi o jeho zdravotním stavu. I přes podporu několika tisíců lékařů a 2,5 milionů užívajících pacientů v roce 2012, Ministerstvo zdravotnictví projekt ukončilo. Podle ministerstva nebyl IZIP dostatečně využíván pacienty ani lékaři. [6]

2.2 Existující přístupy k tvorbě EHR

Existuje několik mezinárodně uznávaných přístupů k tvorbě EHR systémů. Následující výčet přístupů k tvorbě EHR systémů zahrnuje mezinárodně uznávané standardy, ale i modely, které jsou v praxi úspěšně používané.

2.2.1 HL7 (v2, v3)

HL7 (Health Level 7) je sada mezinárodních standardů vytvořených mezinárodní společností pro tvorbu ANSI standardů Health Level Seven International, která byla založená roku 1987. Tato nezisková organizace se zabývá poskytováním komplexního řešení a souvisejících norem pro výměnu, integraci, sdílení a vyhledávání elektronických informací z oblasti medicíny. Tyto standardy definují, jak by se medicínské informace měly ukládat a jak by s nimi měly pracovat různé systémy. Definují různé struktury a datové typy pro ukládání dat a bezpečnou integraci mezi systémy. Standardy jsou zaměřené hlavně na aplikační vrstvu respektive sedmou vrstvu modelu OSI (Open System Interconnection), který vytvořila organizace ISO pro standardizaci počítačových sítí. HL7 standardy jsou uznávány jako vhodné pro praktické použití na světě. [7] [8]

Mezi hlavní standardy patří standard HL7 v2 a HL7 v3. Dalšími pomocnými standardy jsou CDA (Clinical Document Architecture), CCD (Continuity of Care Document), SPL (Structured Product Labeling) a CCOW (Clinical Context Object Workgroup). Standardy HL7 v2 a v3 definují schopnosti různých lékařských a zdravotnických systémů vzájemně si poskytovat služby a efektivně spolupracovat. [9]

HL7 v2 messaging standard definuje elektronickou výměnu dat mezi systémy v lékařském prostředí. Jedná se pravděpodobně o nejvíce implementovaný standard pro eHealth na světě. Tento standard definuje výměnu lékařských dat mezi elektronickými systémy. Zaměřuje se na poskytovatele zdravotní péče a zároveň na vývojáře informačních technologií ve zdravotnictví. HL7 v2 definuje sadu elektronických zpráv ke komunikaci mezi administrativními, logistickými a zdravotními systémy, ale i komunikaci mezi téměř všemi institucemi a oblastmi zdravotní péče. Standard má několik modifikací, z nichž verze 2.3 je nejstarší běžně používanou verzí standardu, která byla schválená jako ANSI standard v roce 1997. Verze HL7 v2 je v současné době stále podporována a rozšiřována paralelně s normou HL7 v3. [10]

HL7 v3 messaging standard je založený na referenčním informačním modelu RIM (Reference Information Model), který HL7 sama vytvořila. Standard definuje, jak by měly být aplikace vyvíjeny, a specifikuje metodiku vývoje, což nutí vývojáře nejprve analyzovat a plánovat architekturu svých rozhraní a aplikací. RIM poskytuje jasné vztahy mezi různými informacemi. Standard HL7 v3 dále definuje úplnou sadu zpráv, datové typy a terminologii pro práci s daty. Tento standard využívá pro práci s informacemi model řízené metodiky, která reprezentuje informace pomocí XML syntaxe. Oproti HL7 v2 je HL7 v3 založen na objektově orientované metodologii. Dále se zaměřuje na celý zdravotnický systém, nejen na nemocniční prostředí, jak tomu bylo u HL7 v2. Specifikuje úplný kontext medicínských informací, což zajišťuje sdílení významu při přenášení informací mezi různými systémy. [11]

2.2.2 CEN/ISO EN13606

CEN/ISO EN13606 je evropský standard vytvořený evropským výborem pro normalizaci (CEN). Je také schválený mezinárodní organizací pro normalizaci (ISO). Byl vytvořen pro zajištění interoperability mezi EHR systémy v komunikaci. Standard vytváří robustní a stabilní informační architekturu EHR systémů, která umožňuje snadnou komunikaci mezi systémy, nebo mezi EHR systémy a centrálním EHR úložištěm (např. elektronická zdravotní dokumentace). Standard je také využíván pro komunikaci EHR systémů a ostatními zdravotnickými aplikacemi, nebo jinými zdravotnickými zařízeními, které pracují s EHR daty. [12]

CEN/ISO EN13606 používá inovativní architekturu oddělení modelu pro informace a znalosti. Informace definuje referenční model, který je tvořen základními prvky reprezentující informace potřebné pro kompletní popsání EHR dat, podobně jako u HL7 v3. Referenční model tedy definuje ukládání EHR dat. Model znalostí je založený na archetypech, které formálně popisují zdravotnické koncepty, jako například měření krevního tlaku. Model archetypů definuje sémantický popis dat referenčnímu modelu. [12]

Referenční model reprezentuje generické a stabilní vlastnosti zdravotních informací. Tento objektově orientovaný model obsahuje malou množinu tříd, které popisují elementární složky EHR záznamů. Dále model definuje pomocné třídy k popisu kontextu EHR. Model obsahuje také třídy k popisu demografických dat a ke komunikaci EHR fragmentů. [13]

Archetypy znalostního modelu obsahují elementy z referenčního modelu, které reprezentují určitý zdravotnický záznam. Tyto archetypy vytvářejí zdravotničtí odborníci. Archetypy by měly obsahovat hlavičku, definici a ontologii. Hlavička obsahuje informativní data o objektu, který archetyp definuje. V definici archetypu jsou entity z referenčního modelu definující zdravotnický koncept. Ontologie obsahuje popis entit z definice a jsou navázány na termíny z nějaké terminologie. [14]

2.2.3 OpenEHR

OpenEHR je nadace, která pracuje na integraci elektronických zařízení a vzájemné spolupráci různých systémů ve zdravotnictví. Dále se zabývá zacházením s elektronickými zdravotními informacemi o stavu pacienta. [15]

Nadace OpenEHR zveřejnila sadu specifikací popisující referenční model pro práci se zdravotnickými informacemi, dotazovací jazyk a jazyk pro popis archetypů a dalších zdravotnických modelů (šablony). Architektura OpenEHR je vytvořena pro použití s libovolnou externí terminologií, jako je například LOINC a ICDx nebo SNOMED CT. Specifikace není zaměřena na řešení komunikačních úkolů ve zdravotnictví a mezi zdravotnickými systémy. Tímto se zabývají výše zmíněné HL7 standardy. [15]

OpenEHR využívá vícevrstvé (multi-level) modelování uvnitř softwarové architektury orientované na služby. V této architektuře vytvářejí různí odborníci modely na příslušných vrstvách. Tito odborníci, jako například lékaři a specialisté ve zdravotnických oborech, jsou přímo zapojováni do definování sémantického popisu zdravotnického informačního systému. Tyto modely, takzvané archetypy, a jejich specifikace je standard ISO (ISO 13606-2 viz kapitola CEN/ISO EN13606). Tyto ISO standardy jsou používány k definování různých národních e-health informačních systémů. OpenEHR přístup umožňuje oddělený vývoj back-end a front-end komponent EHR systému díky zdravotnickým informačním modelům, datovým typům a různým druhům rozhraních služeb, které openEHR definuje. [15]

Nad daty musí být definovány zdravotnické procesy. OpenEHR je definuje prostřednictvím EHR informačního modelu. Jedná se o koncept stavového automatu s instrukcemi a akcemi, který popisuje standardní model zdravotnických procesů. Tento instrukční stavový automat slouží k mapování kroků od konkrétních intervenčních

pracovních postupů jako je například očkování nebo předepisování léků. Elementární akce jsou používány pro vytváření vyšších vrstev konceptu. [15]

Archetypy jsou počítačově zpracovatelné fundamentální specifikace medicínských informací, připravené pro sběr dat. Sbíraná data jsou pak strukturovaná dle modelu archetypu. Hlavní výhoda archetypů je, že jsou definovány pro opakované použití a jsou separované z funkčního softwaru. Často definují výstupní informace z rutinních zdravotnických vyšetření, ale mohou definovat i velice specializované zákroky. Archetypy jsou součástí zdravotnického modelu architektury OpenEHR. [15] [16]

Šablony jsou dalším prostředkem pro tvorbu zdravotnických modelů. Tyto šablony openEHR definuje pomocí formátu OET, což je jazyk pro formální popis šablon. Mohou se skládat z jednoho nebo více archetypů. Mohou se skládat i z jejich částí. Šablony jsou určeny pro použití v konkrétních případech, kde není možné použít striktně jeden celý archetyp. Například archetypy pro měření krevního tlaku, hmotnosti a cukru v krvi mohou být spojeny do šablony, která bude použita pro diabetické vyšetření nebo pozorování vývoje cukrovky. [16]

2.3 Terminologie

Terminologie slouží k sémantické definici odborných termínů. Používání jednotné terminologie slouží k efektivní komunikaci mezi odborníky. Terminologií bývá označen i celý systém sloužící ke specifikaci odborných termínů. Následující tři terminologické systémy se zabývají medicínskými termíny.

2.3.1 SNOMED CT

SNOMED CT je jedna z nejobsáhlejších, nejpřesnějších a nejpoužívanějších zdravotních terminologií na světě. Je vedena a spravována organizací IHTSDO (The International Health Terminology Standards Development Organization). Byla vyvinuta pro použití a potřeby zdravotnictví na celém světě. Jeho obsah je vědecky ověřován, a je tedy důvěryhodný. Poskytuje konzistentní, počítačově snadno zpracovatelnou reprezentaci zdravotnických termínů, které jsou v souladu s mezinárodními standardy platných ve většině zemí světa. Terminologie SNOMED CT je používán ve více než padesáti zemích světa. [17]

Je založen na konceptu unikátních zdravotnických termínů a formálních definic, které jsou organizovány v hierarchii. Obsah SNOMED CT je reprezentován termíny, popisy a vztahy mezi termíny. [18]

Termín reprezentuje zdravotnický výraz, který má unikátní číselný identifikátor. Uvnitř hierarchické struktury termínů jsou termíny organizovány od nejobecnějších k detailním. Tvoří tak komplexní stromovou hierarchii na základě vztahů mezi termíny. [18]

Ke každému termínu je přiřazen popis, který je lidsky interpretovatelný. Termín může mít i více popisů. Popis může být i synonymum původního termínu. Každý popis má unikátní číselný identifikátor. [18]

Poslední komponentou jsou vztahy. Ty spojují termíny, které jsou si příbuzné nebo podobné v nějakém smyslu. [18]

Terminologie bývá součástí nějaké aplikace, která využívá její výhody. V případě použití v EHR systémech vylepšuje komunikaci a zvyšuje přístupnost důležitých medicínckých informací. SNOMED CT je užitečná pro zdravotní dokumentaci jako podpora reprezentace detailních zdravotních informací. [19]

2.3.2 LOINC

LOINC je jazyk pro medicínské nebo laboratorní testy, měření a pozorování. Definuje množinu identifikátorů, jmen a kódových značek. LOINC je možné jednoduše použít jako terminologii. Autor Clem McDonald vytvořil LOINC v roce 1994 jako vyšetřovatel institutu Regenstrief. Tato mezinárodně uznávaná nezisková organizace se zaměřuje na vývoj běžných terminologií pro zdravotnické a laboratorní účely. Terminologie LOINC začala být využívaná pro zasílání zdravotnických dat z laboratoře do nemocnice elektronickou formou. [20]

Mnoho zdravotnických zařízení používá k elektronické výměně zdravotnických informací HL7 standardy. V různých systémech jsou tyto informace identifikovány pomocí jejich interních kódových značek. Po odeslání zprávy jedním systémem, pak systém, který zprávu přijme, nemusí zprávě plně rozumět a náležitě ji zpracovat, aniž by přijal kódové značky odesílatele zprávy. To je při přijímání zpráv z více zdrojů nemožné.

Tento problém řeší LOINC, který poskytuje univerzální kódové značky a jména, a tvoří tak globální jazyk pro identifikování různých zdravotnických úkonů. [20]

2.3.3 DASTA

DASTA je otevřený standard podporovaný Ministerstvem zdravotnictví České republiky v roce 1997. Tento standard slouží pro komunikaci informačních zdravotnických systémů a sdílení dat mezi těmito systémy v rámci České republiky. Hlavním součástí tohoto standardu je více než tři sta číselníků, které přiřazují termínům číslo. Tyto číselníky zahrnují např. termíny pro Národní zdravotnický informační systém (NZIS) a pro laboratorní obory NČLP (Národní Číselník Laboratorních Položek), což je jediný komunikační standard v této oblasti. Další součástí tohoto standardu je program pro práci s číselníky. [21]

3. MOBILNÍ APLIKACE

V dnešním světě jsou mobilní zařízení součástí běžného života lidí. Aplikace na těchto zařízení mohou jejich uživatelům usnadnit a zpříjemnit život. Instalací nové mobilní aplikace se mobilnímu zařízení předá nová funkce. K těmto aplikacím patří například hry, překladače textu, ovládání telefonu hlasem, navigace, atd. Většina uživatelů těchto aplikací využívá již hotové aplikace. Někteří si je však mohou vytvořit, popřípadě si je mohou nechat vytvořit, pro řešení jejich konkrétních požadavků, které jim usnadní život. [22]

3.1 Porovnání platforem

V této kapitole jsou porovnávány platformy Android, iOS a Windows Phone. V současné době všechny tři platformy podporují běh více aplikací zároveň (multitasking), což umožňuje aplikacím, které běží na pozadí provádět nějakou činnost jako např. synchronizace dat se vzdáleným serverem. Z pohledu vývoje mobilních aplikací je podstatné, že pouze Android je Open source. Další srovnání je uvedeno v tabulce (viz Tabulka 1).

	Android	iOS	Windows Phone
Multitasking	ano	ano	ano
Jádro	Monolitické (Linux)	Hybridní jádro	Hybridní jádro (Windows NT) Od verze Windows Phone 8
open source	ano	-	-
OTA aktualizace	ano	ano	ano
Podíl na trhu 2. kvartál v roce 2015	82,80%	13,90%	2,60%
Průměrná cena zařízení s OS v roce 2015	$250-$300	$600	$150
Zdroj aplikací	Windows store	Google play	App Store
programovací jazyk	Java	Swift	C#, C++
Vývojové prostředí	Android Studio	Xcode	Visual Studio

Tabulka 1 Porovnání mobilních platforem

Zdroj: vlastní

V řádku, který popisuje podíl na trhu ve druhém kvartálu v roce 2015, je vidět, že dominují na trhu mobilní zařízení s operačním systémem Android a iOS. Rozdíl těchto platforem je také finanční stránka vývoje aplikací, kde projekty pro Android bývají nákladnější než u iOS. Ačkoli je na Android stahováno více aplikací, výnos ze stažených

aplikací je nižší než u iOS. To může být dáno tím, že pro Android je daleko více neplacených aplikací nebo jsou uživatelé mobilních zařízení s operačním systémem iOS ochotnější využívat placené aplikace. [23]

3.2 Android

Jedná se o rozšířený mobilní operační systém založený na linuxovém jádře. Tento operační systém je vytvořen společností Google a je dostupný jako open source. Aplikace mohou využívat funkce jádra operačního systému pomocí Android API. [24]

3.2.1 Specifikace

Android byl původně určen pro mobilní platformy a jejich progresivní rozvoj s nízkými náklady na vývoj a distribuci. Dnes je tento operační systém používán nejen na mobilních platformách, ale i např. v chytrých televizích a autech. Byl postaven tak, aby vývojářům umožnil vytvářet inovativní mobilní aplikace s využitím všech vlastností daného mobilního zařízení, jako například obsluha telefonních hovorů, práce s textovými zprávami, nebo využívání fotoaparátu. Systém je postavený na monolitickém otevřeném jádře Linux a používá vlastní virtuální stroj (Dalvik do verze 4.3, ART od verze 4.4 včetně), speciálně navržený pro optimalizaci práce s pamětí a hardwarovými prostředky, především k poměru úspory energie a výkonu. [24]

3.2.2 Java jazyk

Android aplikace jsou napsané v objektově orientovaném programovacím jazyce Java. Mohou využívat většinu knihoven Java SE (Standard Edition). Hlavním rozdílem je nepřítomnost knihoven pro uživatelské rozhraní. Tyto knihovny byly nahrazeny speciálně upravenými knihovnami pro Android uživatelské rozhraní. [25]

Velkou výhodou Javy je její přenositelnost mezi různými platformami. Tato výhoda velice usnadňuje vývoj aplikací pro co nejširší okruh mobilních zařízení. Hotový zdrojový kód se převádí do bytecode, se kterým virtuální stroj androidu umí pracovat. U virtuálního stroje Dalvik se překlad prováděl pomocí stejného kompilátoru jako v případě běžných Java

aplikací do Java bytecode a následně ho Dalvik kompilátor převedl do Dalvik bytecode. Takto zpracovaný zdrojový kód je zabalený do balíčku systému android – souboru APK. [26]

Kromě standardního Java SE JDK (Java Development Kit) je potřeba k vývoji android aplikací také sada Android SDK (Software Development Kit), která poskytuje nástroje pro vývoj, debugging a testování. Android SDK obsahuje také komponenty specifické pro různé verze androidu. Je tak možné otestovat funkčnost aplikace na různých verzích androidu. Důležitou součástí Android SDK je také emulátor, který umožňuje ladění aplikací bez přítomnosti fyzického zařízení. [26]

3.2.3 Android studio

Android studio je oficiální vývojové prostředí aplikací pro android. Prostředí je založené na IntelliJ IDEA od společnosti JetBrains Inc. pro vývoj Java aplikací. Prostředí splňuje všechny vlastnosti moderního vývojového prostředí se všemi jeho nástroji pro vývoj, debugging a testování. Android studio používá buildovací nástroj Gradle, který je přímo zabudovaný v Android studiu. Toto vývojové prostředí má velice povedený nástroj pro tvorbu uživatelského rozhraní. Komponenty lze umisťovat do okna aplikace pouhým výběrem a vložením na danou pozici pomocí myši. Každá šablona grafického uživatelského rozhraní je definována XML souborem. Je tedy možná i přímá tvorba a úprava uživatelského rozhraní pomocí editace tohoto souboru. [27]

4. DATOVÁ ÚLOŽIŠTĚ

V mobilních aplikacích je často nutné ukládat data, které aplikace ke svému chodu používá. Tato data musí být dostupná i po ukončení aplikace a jejím opětovném spuštění. K jednoduchému ukládání a načítání dat slouží databázové systémy, které na mobilních platformách fungují jinak než klasické systémy řízení báze dat (SŘBD).

4.1 Porovnání relačních a NoSQL databází

Ačkoliv tradiční relační databáze existují již řadu let a jsou stále vylepšovány, při práci s velkým objemem nestrukturovaných dat nejsou příliš efektivní. Struktura dat v relačních databázích je předem definovaná strukturou tabulek, relacemi mezi tabulkami a je závislá na jménech a datových typech sloupečků v těchto tabulkách. [28]

V posledních letech se objevila potřeba ukládat velké množství dat z různých zdrojů, jako jsou například sociální media, mobilní a IoT (Internet of Things) aplikace. K používání relačních databází bývají zapotřebí servery s výkonným hardware. Rozšíření těchto databází znamená distribuovat databázi na více serverů a práci s takovou databází není jednoduchá. Relační databáze nabízí spoustu funkcí pro práci s daty a datovou integritu. Většinu těchto funkcí běžný uživatel databáze ani nevyužije a navíc zvyšují složitost a cenu databáze. [28]

Z důvodu těchto omezení relačních databází se v posledních letech začínají stále více používat NoSQL nerelační databáze. Hlavní výhodou těchto databází je flexibilní datový model, který neurčuje striktní databázové schéma. Toto schéma určují aplikace používající tuto databázi. Umí proto efektivně pracovat s nestrukturovanými daty jako jsou multimedia, data ze sociálních medií, atd. Většina NoSQL databází je open source. Pro práci s tímto typem databáze není nutná znalost jazyka SQL. Některé NoSQL databáze umožňují flexibilní škálovatelnost, respektive jednoduchou distribuci databázového systému na více serverů (klastrů). To zajišťuje vyšší bezpečnost a dostupnost dat a relativně levnou rozšiřitelnost databázového systému. [28]

4.2 Relační databáze pro mobilní platformy

V této kapitole je popsána knihovna pro řešení databázového systému založeného na relačním modelu, která se běžně používá na mobilních platformách. Tato knihovna funguje v různých modifikacích pro mobilní platformy Android, iOS a Windows Phone.

4.2.1 SQLite

Jedná se o relační databázový systém, který je obsažen v malé knihovně. Tuto knihovnu je možné používat v různých implementacích pro Android, iOS i Windows Phone. Pro použití v mobilních zařízeních je ideální, protože je velice úsporný k paměťovým zdrojům a je relativně rychlý. SQLite nefunguje na principu klient-server jako jiné relační databázové systémy, které jsou spuštěné jako samostatné procesy. SQLite funguje jako knihovna v rámci každé aplikace, která jej používá. Knihovna se v případě potřeby jednoduše připojí k aplikaci. Je možné ji používat prostřednictvím jednoduchého rozhraní. Další velká výhoda této knihovny je také v tom, že je možné jí libovolně použití bez nutnosti jakékoli platby. [24]

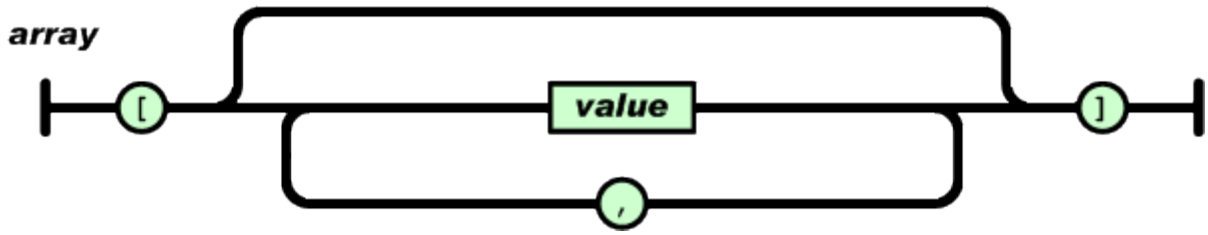
SQLite funguje jako klasická relační databáze využívající dotazovací jazyk podobný SQL. Pro použití na mobilních zařízeních musel být jazyk trochu ochuzen. Nelze například vložit do tabulky cizí klíč, neexistuje datový typ DATE ani typová kontrola při vkládání dat a příkaz ALTER TABLE je také omezený. Tato omezení nutí vývojáře uzpůsobit svojí aplikaci těmto omezením. [24]

4.3 NoSQL databáze

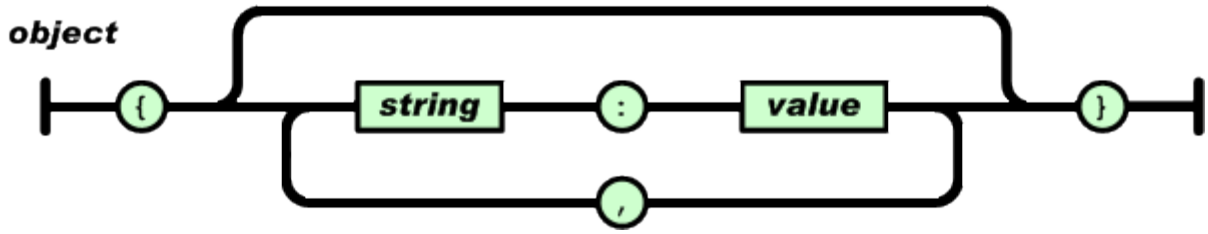
Existuje několik typů NoSQL databází. První typ je uložení klíč-hodnota. Jedná se o systém ukládání hodnot, které jsou indexované klíči pro jejich zpětné získání. Tyto systémy jsou schopné uchovávat strukturovaná i nestrukturovaná data. Druhým typem je sloupcově orientovaná databáze. Tato databáze obsahuje jeden rozšiřitelný sloupec, do něhož ukládá spolu související data. Dalším typem jsou dokumentově orientované databáze. Tyto databáze ukládají a organizují data jako kolekce dokumentů. V dokumentech může být různý počet položek s různou velikostí. Existují také další NoSQL databáze jako např. grafová, nebo hybridní. [28]

4.3.1 Couchbase

Couchbase nabízí několik produktů pro databázové řešení webů, mobilních a dalších aplikací. Hlavním produktem je Couchbase Server, NoSQL dokumentově orientovaná open source databáze. Dokumenty v této databázi jsou reprezentovány JSON (JavaScript Object Notation) dokumenty, které obstarají indexování dat pro jejich rychlé dotazování. Datový model založený na JSON dokumentech je velice flexibilní. JSON ukládá data jako text, který je jednoduše lidsky čitelný i strojově zpracovatelný. JSON dokáže ukládat strukturu polí (viz Obrázek 1), objektů (viz Obrázek 2) a jejich kombinací. Schéma databáze je vytvářeno v kódu aplikace, respektive strukturou dokumentů, které aplikace do databáze vkládá. To velice usnadňuje práci vývojářům, kteří mohou do databáze vkládat dokumenty s libovolnou strukturou, aniž by museli upravovat datové schéma na databázové úrovni. [29]



Obrázek 1: Schéma tvorby JSON polí.
Zdroj: http://www.json.org/object.gif



Obrázek 2: Schéma tvorby JSON objektů.
Zdroj: http://www.json.org/array.gif

Pro přístup, změny a manipulaci s daty nabízí Couchbase server vlastní jazyk N1QL založený na SQL. Tento jazyk podporuje např. JOIN operace pro spojování, což není běžné u NoSQL databází. [29]

Dalším produktem je Couchbase Mobile. Jedná se o NoSQL databázové řešení pro mobilní zařízení. Databáze je schopna fungovat se síťovým připojením i bez něj. Mobilní aplikace používající tuto databází mají stálý přístup k datům. Data se ukládají do lokální

databáze na mobilním zařízení. Když má mobilní zařízení přístup k síti je možné lokální databázi synchronizovat se vzdálenou databází Couchbase server. Couchbase Mobile se skládá ze dvou komponent. [30]

První je Couchbase Lite, což je objektově orientovaná NoSQL databáze ukládající data v mobilním zařízení. Běží vždy v rámci nějaké aplikace. Používá MapReduce pro upozornění o změně v datech a synchronizaci se vzdáleným databázovým serverem. [30]

Druhou komponentou je Sync Gateway. Tato komponenta obstarává synchronizaci dat se vzdáleným databázovým serverem. Synchronizace je nastavována uvnitř mobilní aplikace, která využívá Couchbase Lite. Sync Gateway také autentizuje a autorizuje uživatele, který používá aplikaci. [30]

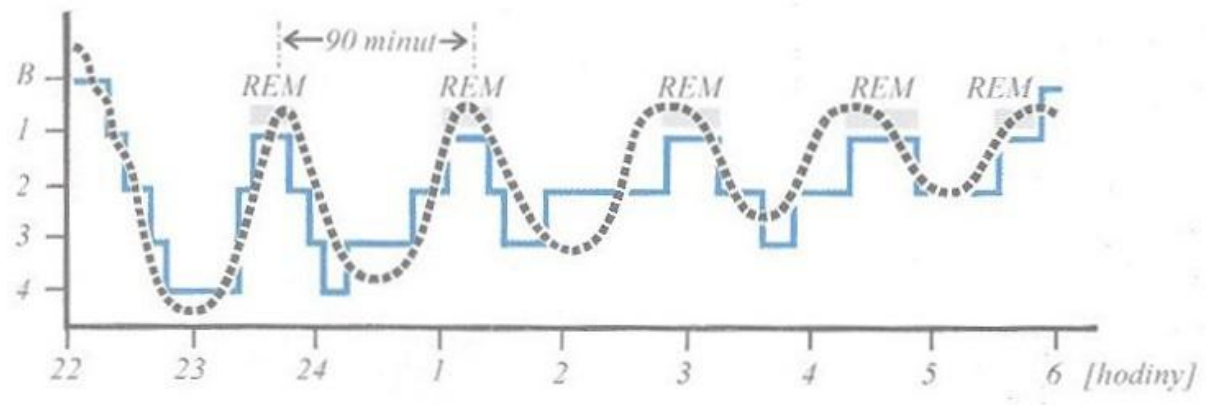
4.3.2 Elasticsearch

Elasticsearch je moderní dokumentově orientovaná NoSQL databáze. Data v této databázi jsou uloženy jako strukturované JSON dokumenty. Všechna pole jsou indexovaná, to zajišťuje vysokou rychlost fulltextového vyhledávání. Fulltextové vyhledávání je založeno na Apache Lucene, což je open source knihovna pro rychlé fulltextové vyhledávání napsaná v jazyce Java. Elasticsearch lze jednoduše používat pomocí RESTful API. Toto API používá JSON zprávy zasílané pomocí HTTP protokolu. Díky tomu je možné využívat Elasticsearch v aplikacích napsaných v různých jazycích. JSON zprávy umožňují jednoduché a rychlé dotazování a filtrování dat. Elasticsearch umožňuje velice jednoduchou rozšiřitelnost na více serverů. To zajišťuje bezpečnost a jednoduše rozšířitelnou velikost uložených dat. [31]

5. SPÁNEK

Spánek je stav organismu, který se vyskytuje u všech vyšších obratlovců. Opakem spánku je stav bdělosti, v němž se projevuje různé chování. Spánek lze charakterizovat jako stav klidu a relaxace. [32]

Spánek je dynamický proces, ve kterém se střídají různé úrovně hloubky. Opakuje se každodenně a i jeho vnitřní struktura je cyklická. Spánek se dělí do dvou fází, které se v jeho průběhu mění (viz Obrázek 3). První fáze je spánek hluboký nebo takzvaně non-REM (NREM – Non Rapid Eye Movement). Druhá fáze je spánek rychlý. Tato fáze se vyznačuje rychlými očními pohyby (REM spánek – rapid eye movement). V dospělosti převažuje fáze spánku NREM v poměru 4:1. Naopak u novorozenců převažuje fáze REM, která zastupuje až 80% spánku. Tento poměr však do dvacátého roku života postupně klesá. Průměrný spánkový cyklus člověka, ve kterém se střídají spánkové fáze, se pohybuje od 90 do 100 minut. To je však pouze průměr, extrémy se mohou pohybovat v rozsahu od 20 do 170 minut. Během celého spánku se může vystřídat 4 až 6 spánkových cyklů. Cykly uprostřed spánku bývají delší než cykly na začátku a na konci spánku. [33] [32]



Obrázek 3 Spánkové cykly – hypnogram
Zdroj: [32]

Předpokládá se, že pomalý, hluboký spánek slouží k regeneraci somatických funkcí a rychlý spánek k obnově mozkových funkcí. V NREM spánku se zpomaluje srdeční činnost, tím pádem se snižuje spotřeba kyslíku. Dále se mění systolický krevní tlak, zpomaluje se dýchání a klesá tělesná teplota. Naopak v REM spánku se srdeční činnost zvyšuje, krevní

tlak roste a dýchání je velmi nepravidelné. V organismu vzniká takzvaná „vegetativní bouře“. [32]

K monitorování spánku slouží polygrafický záznam, který je pořizován pomocí polygrafu. Jedná se o lékařské zařízení, které sleduje několik fyziologických funkcí člověka. Sleduje například saturace krve kyslíkem, krevní tlak, průběh dýchání, mozkovou a srdeční činnost. Toto zařízení je využíváno také v kriminalistice jako takzvaný „detektor lži“, ale i v mnoha dalších nelékařských oborech. Před měřením se na pacienta připevní všechna potřebná čidla. Samotné měření je možné provádět v laboratoři, ale i doma. Záznam z tohoto zařízení má podobu tabulky nebo grafu [34].

5.1 Fáze spánku REM

Tuto fázi nazýváme také jako spánek paradoxní, desynchronizovaný nebo rombencefalický. V této fázi spánku se fixuje paměťová stopa. To znamená, že zejména v REM fázi spánku se posiluje paměť. Teoreticky to může znamenat, že člověk, který se učí před spaním, si uchová a následně lépe vybaví více informací než člověk, který se učí v jinou denní dobu. REM fáze spánku se vyznačuje fyzickými změnami na elektroencefalogramu (zkráceně EEG), označované jako ponto-genikulo-okcipitální hroty (PGO spikes). Samovolný vznik hrotů v místech průběhu zrakové dráhy svědčí o zvyšující se aktivitě očí. Proto je tato fáze charakterizována rychlými očními pohyby (REM – rapid eye movement). Na EEG záznamu spánku je v této fázi vidět nízká amplituda a desynchronizace. Na rozdíl od ostatních fází spánku zde nejsou dominantní žádné mozkové vlny. V této fázi spánku je velmi lehké spícího vzbudit, je to lehký spánek. Tato fáze je také typická pro snění. [32] [33]

5.2 Fáze spánku NREM

Nazýváme ho také spánek pomalý, synchronizovaný, s pomalými vlnami, ortodoxní nebo také telencefalický. V této fázi se rychlé oční pohyby již neobjevují. Mozková aktivita se v průběhu NREM spánku utlumuje. V této fázi je také největší sekrece růstového hormonu. Fáze NREM se dělí na 4 stádia. [32] [35]

Prvním stádiem je NREM 1. Je to první fáze spánku takzvané usínání. Jedná se o lehký spánek, kdy se člověk může probudit i na slabý podnět. V této fázi nejsou rychlé oční pohyby, nicméně se mohou objevovat pomalé oční pohyby. Mozková činnost typická

pro bdění se zavřenýma očima, takzvaná α-aktivita (8 – 13 Hz), se pozvolna vytrácí a objevují se vlny theta (4 – 7 Hz). [32] [33]

Druhým stádiem je NREM 2. V tomto stádiu se theta vlny stupňují a začínají se objevovat spánková vřetena a K-komplexy (potenciál se třemi fázemi následovaný vřetenovitou aktivitou o frekvenci kolem 13 Hz). Je možné člověka probudit například oslovením. Toto stádium zaujímá 45% - 55% celého spánku. Dále se zde snižuje svalový tonus a v tomto stádiu člověk ztrácí vědomí. [32] [33]

Třetím stádiem je NREM 3. V tomto stádiu se objevují velice pomalé vlny s frekvencí 0,5 – 4 Hz, které nazýváme vlny delta. Spánková vřetena a K-komplexy se zde vytrácejí. V tomto stádiu je již spící ve hlubokém spánku a je relativně obtížné spícího probudit. Může se objevovat náměsíčnost a mluvení ze spaní. [32] [33]

Posledním stádiem hlubokého spánku je NREM 4. V tomto stádium dominují z více než 50% pomalé vlny delta. Proto je toto stádium označováno jako δ-spánek. Toto stádium je nejčastější v prvních několika cyklech spánku. Jedná se již o velmi hluboký spánek, kdy můžeme vzbudit spícího teprve na bolestivý podnět. Toto stádium slouží pravděpodobně k upevňování deklarativní paměti. [32] [33]

5.3 Spánkové aplikace

Současné spánkové aplikace pracují na téměř stejném principu. Využívá se akcelerometr telefonu, který monitoruje pohyby spícího v průběhu spánku. Na základě získaných informací aplikace vyhodnotí, ve které fázi spánku se spící nachází. Když se spící v době vstávání nachází ve fázi lehkého spánku, aplikace spícího probudí. Pokud by spící nebyl v intervalu ani jednou ve fázi lehkého spánku, aplikace ho vzbudí na konci intervalu. Aplikace se liší vzhledem a dalšími funkcemi.

5.3.1 SleepAsAndroid

Tato aplikace je dostupná pouze pro operační systém Android. Byla vytvořena společností Urbandroid Team. Na webu Google Play jsou k dispozici dvě verze. První je placená a podporuje veškeré funkce aplikace. Druhá podporuje veškeré funkce pouze na 14

dní. Po uplynutí této doby se uzamkne většina funkcí a aplikace funguje pouze jako chytřejší budík. [36]

Aplikace v průběhu spánku sleduje a ukládá spánkové cykly uživatele. V době vstávání aplikace čeká, až bude uživatel ve fázi lehkého spánku a poté uživatele probudí. Aplikace dále umí ukládat a zobrazovat historii spánkových grafů, počítat spánkový deficit, sdílet informace na sociálních sítích, speciální zvuky pro buzení (např. zvuky přírody), postupně zvyšovat hlasitost budíku, nahrávání zvukové stopy při chrápání nebo mluvení ze spaní atd. SleepAsAndroid má spoustu dalších uživatelsky zajímavých funkcí, jako je například takzvaná CAPTCHA. Tato funkce vypne budící alarm až tehdy, když uživatel například vypočítá nějaké příklady, dostatečně zatřese telefonem, nebo přiloží telefon ke QR kódu. Následně je budík vypnut. [36]

5.3.2 SleepBot – Sleep Cycle Alarm

SleepBot je jednoduchá spánková aplikace, která je velmi intuitivní. Obsahuje běžné funkce, jako je chytré vstávání při lehkém spánku ve zvoleném intervalu, sledování a ukládání pohybu v průběhu spánku, záznam zvuku nebo připomenutí času, kdy by měl uživatel jít spát. Jako většina spánkových aplikací sleduje dobu strávenou spaním a dobu spánkového deficitu. Z těchto dat může sestavovat různé analýzy a poskytovat různé rady pro zkvalitnění spánku. Aplikace automaticky zálohuje a sdílí data na domovské stránce této aplikace. Aplikace je zdarma ke stažení na webu Google Play pro operační systém Android. Je také dostupná pro operační systém iOS. [37]

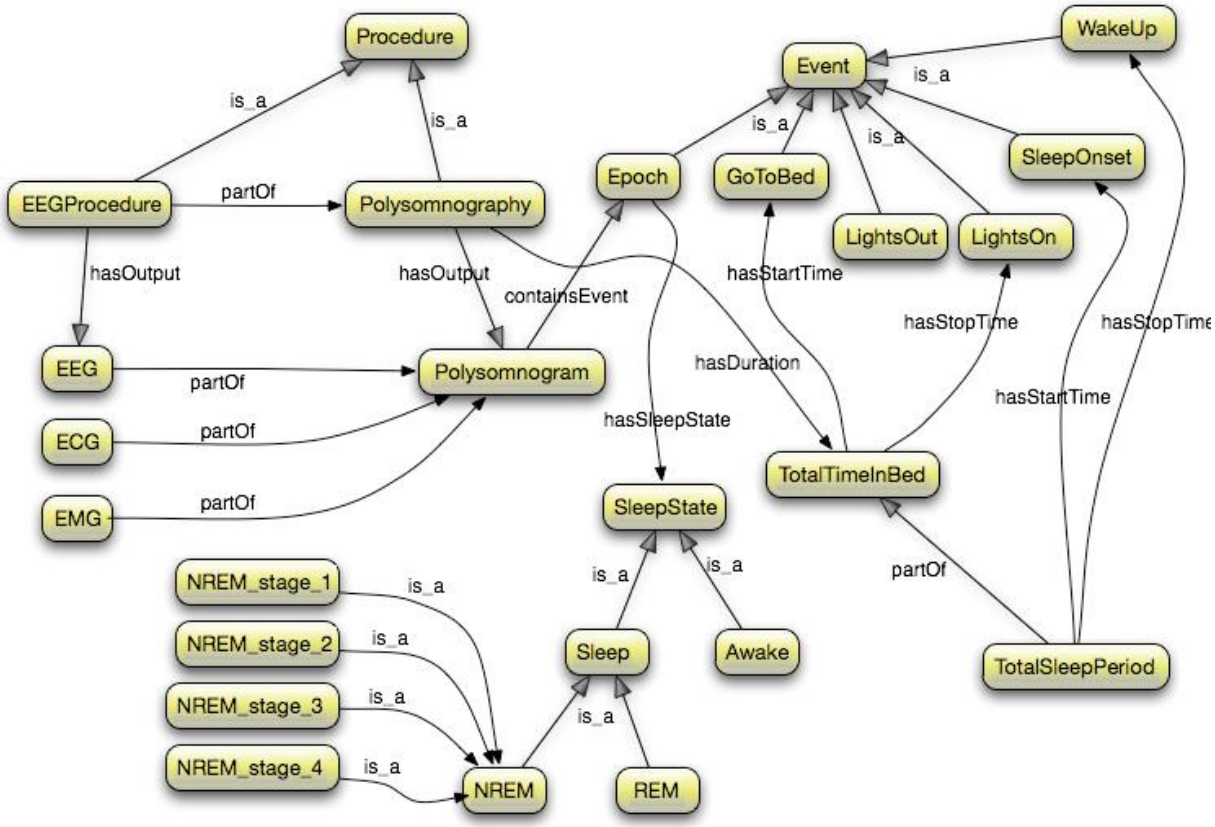
5.3.3 SleepCycle

Tato spánková aplikace existuje ve dvou distribucích. První je pro mobilní telefony s operačním systémem Android a druhý je pro telefony s operačním systémem iOS. V aplikaci je nastavitelná hodnota budícího intervalu. V tomto intervalu aplikace čeká, až bude spící v lehkém spánku a v tu chvíli ho probudí. Spící by se měl probudit odpočatý, jako kdyby vstal samovolným probuzením. Aplikace nabízí funkce jako odložení buzení, zobrazení grafu a statistik průběhu spánku, které můžete poslat emailem nebo sdílet na sociálních sítích, volba budícího zvuku, vibrace při buzení a používání senzoru

vzdálenosti, což umožní vypnutí displeje při položení telefonu displejem dolů. Celá aplikace je velmi jednoduchá a intuitivně se ovládá. [38]

5.4 Spánková ontologie

Pro vytvoření spánkového archetypu, který by formálně popisoval spánek měřený ve spánkových aplikacích, je třeba použít odborné termíny. Tyto termíny musí být podložené ověřenou ontologií. Spánková ontologie je čerpána částečně ze SNOMED CT, kde existuje v rámci této terminologie a částečně ze spánkové ontologie na BioPortal [1], což je repositář celosvětově používaných medicínských ontologií. Tuto ontologii vytvořil doktor Sivaram Arabandi (viz Obrázek 4).



Obrázek 4 Spánková ontologie.
Zdroj: http://bimib.disco.unimib.it/images/7/7c/SSFW09_Arabandi_SleepOntology.pdf

[1] https://bioportal.bioontology.org/ontologies/SDO/

6. NÁVRH ARCHITEKTURY APLIKACE

Rozhodl jsem se pro vývoj aplikace na sběr medicínských dat pro operační systém Android z důvodu dostupnosti, relativně rychlého vývoje a velké rozšířenosti tohoto operačního systému pro mobilní zařízení. Ve své aplikaci používám návrhový vzor MVP (Model-View-Presenter), který je při tvorbě aplikací pro operační systém Android dobře aplikovatelný. MVP slouží jako architektura aplikace pro rozdělení do tří vrstev, což zvyšuje přehlednost a umožňuje jednoduché rozšíření.

Aplikace by měla sloužit jako prototyp pro vývoj mobilní aplikace, která bude pomocí mobilního zařízení sbírat medicínská data uživatele a ukládat je v rámci osobní zdravotní dokumentace na internet. Snadná rozšiřitelnost aplikace byla proto jednou z hlavních priorit při tvorbě tohoto prototypu.

Vývoj aplikace je v rámci projektu openEHR elektronické zdravotní dokumentace, která se vyvíjí na katedře informatiky a výpočetní techniky. Aplikace by měla využívat openEHR archetypy pro formální popis sbíraných medicínských dat. V rámci mé bakalářské práce jsem implementoval sběr spánkových dat. Pro formální popis těchto dat bude potřeba spánkový openEHR archetyp. Data jsou získávána jako vstup od uživatele a z aplikace třetí strany Sleep as Android.

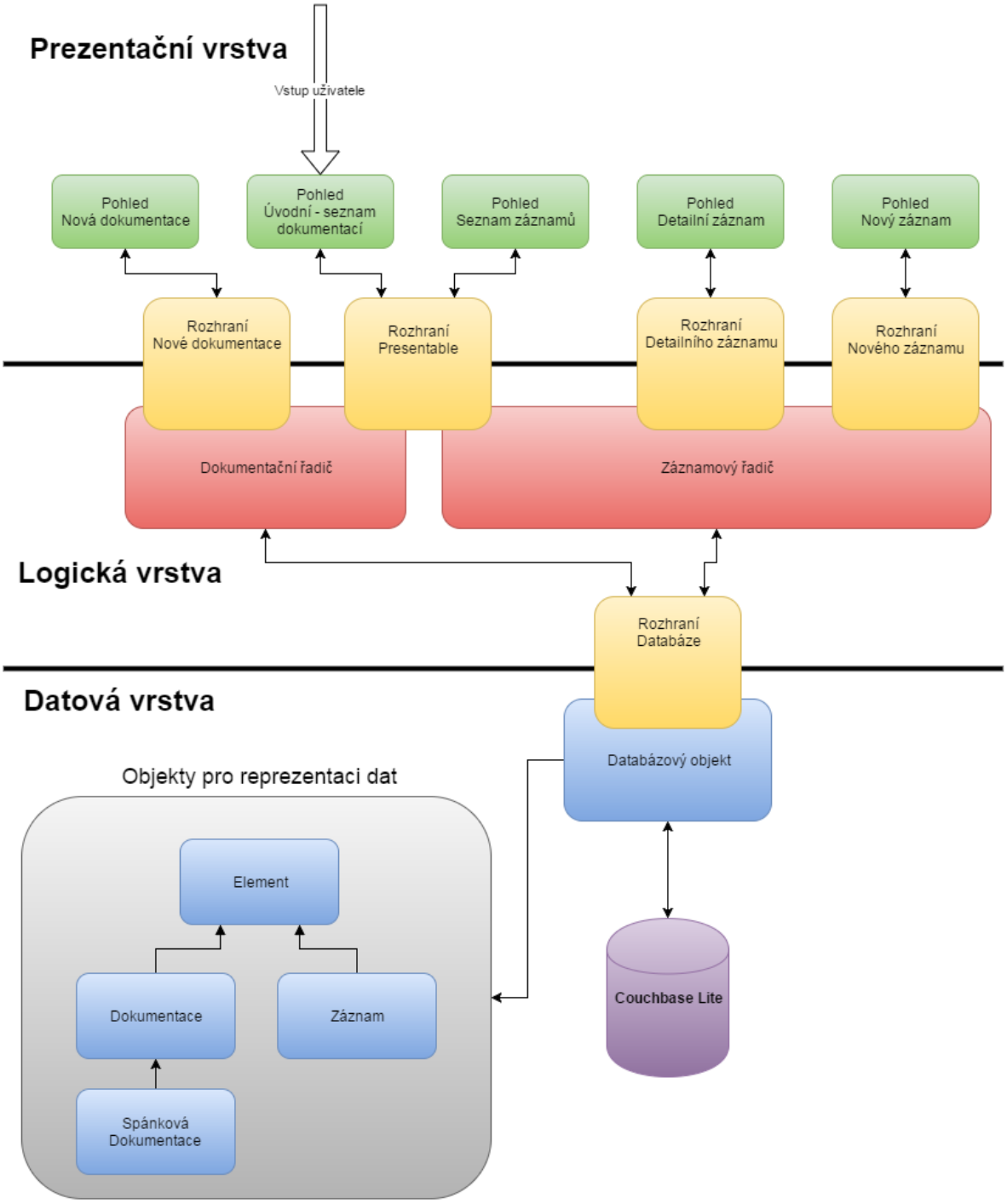
6.1 Třívrstvá architektura Model-View-Presenter

Třívrstvá architektura založená na návrhovém vzoru MVP odděluje model aplikace od uživatelského rozhraní. Uživatel tedy přistupuje pouze k pohledům a přístup k modelu, respektive k datům, obstarává řadič. Pohled dále umožňuje interakci s uživatelem. Dle mého názoru je třívrstvá architektura MVP vhodná pro vývoj mobilních aplikací, z důvodu jednoduché údržby, úpravy a rozšíření kódu a rychlé převedení aplikace na jinou platformu. Celé architektura je zobrazena na obrázku níže (viz Obrázek 5).

V prezentační vrstvě má architektura několik pohledů pro interakci s uživatelem. První pohled zobrazuje seznam všech vytvořených dokumentací. Druhý pohled zobrazuje šablonu pro vytvoření nové dokumentace. Další pohled zobrazuje seznam záznamů z konkrétní dokumentace. Uživatel si dále může nechat zobrazit pohled detailního záznamu nebo pohled vytvoření nového záznamu.

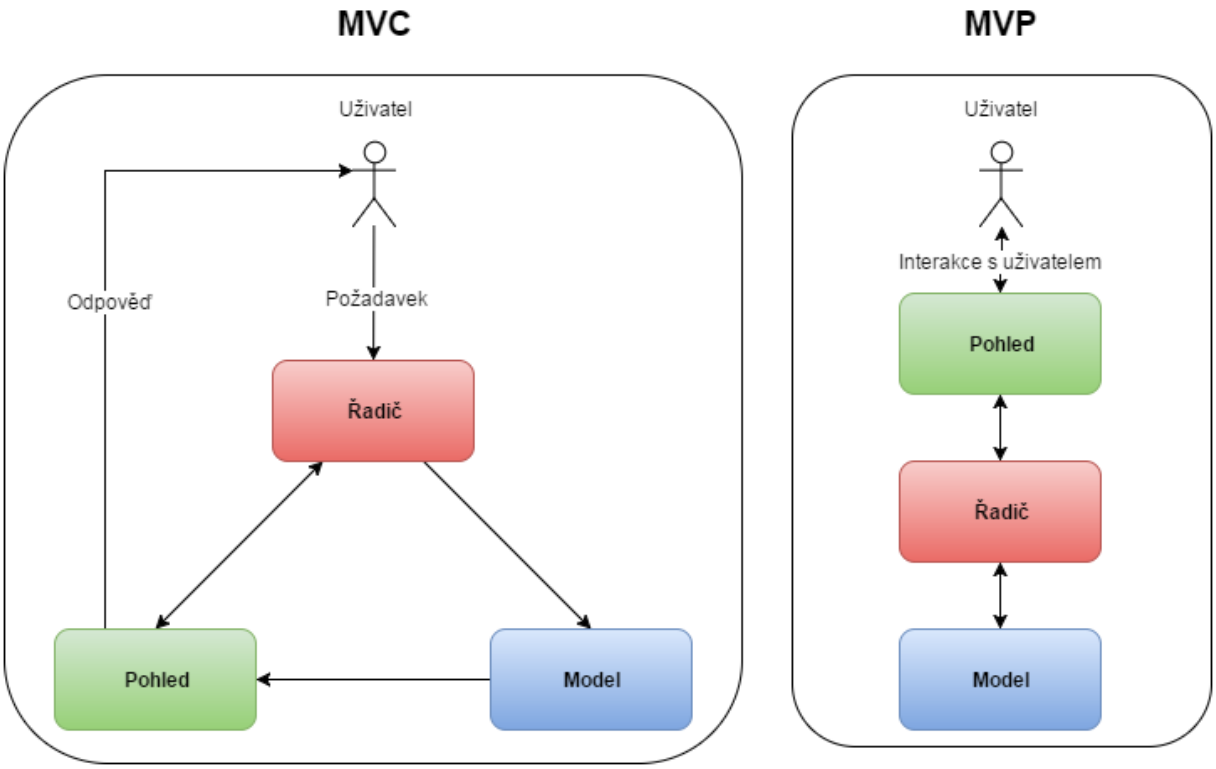
Na logické vrstvě jsou dva řadiče. První řadič obsluhuje události v rámci dokumentací. Druhý řadič obsluhuje události v rámci záznamů konkrétní dokumentace. Pohledy budou komunikovat s řadiči pouze přes příslušná rozhraní.

Na datové vrstvě je databázový objekt, který vytváří, maže, načítá data a vytváří z nich objekty reprezentující datové jednotky. Databázový objekt komunikuje s řadiči přes databázové rozhraní. Řadiče rozhraním komunikují s databázovým objektem a získávají data.



Obrázek 5 Architektura aplikace
Zdroj: vlastní

Oproti MVC (Model-View-Controler) komunikuje u MVP pohled s modelem pouze prostřednictvím řadiče. U MVC může model dodávat pohledu data. U MVP komunikuje uživatel s aplikací pouze prostřednictvím pohledů. K tomuto účelu je vhodné použít Android aktivity, které budou komunikovat s uživatelem a zasílat požadavky řadiči. Tyto aktivity není nutné vytvářet při každém požadavku uživatele. To je v souladu s principem životního cyklu Android aktivit. U MVC obsluhuje interakci uživatele řadič, který zpracuje akci uživatele a odešle mu pohled, který slouží pouze k zobrazení informací. Rozdíly jsou patrné na obrázku níže (viz Obrázek 6).



Obrázek 6 Porovnání MVC a MVP architektury
Zdroj: vlastní

6.1.1 Model

Model zajišťuje práci s daty, jejich uložení a následné načtení z databáze. Na této úrovni pracujeme s elementy, které tvoří datové jednotky. Rozlišujeme dva druhy těchto elementů. Prvním jsou zdravotní dokumentace, které reprezentují skupinu elektronických zdravotních záznamů stejného typu. Druhým jsou jednotlivé záznamy z těchto dokumentací. Každý element má přiřazený jedinečný identifikátor. Tento identifikátor generuje databáze

při uložení elementu. Identifikátor je řetězec, kterým můžeme přistupovat k tomuto elementu v databázi.

Pro práci s databází je vytvořen objekt podle návrhového vzoru jedináček. To znamená, že lze vytvořit pouze jedinou instanci tohoto objektu. Pro práci s databází je to nutné. Je potřeba zajistit, aby k datům přistupovala pouze jedna entita z důvodu konzistence dat. Řadiče používají pro komunikaci s tímto objektem databázové rozhraní, které databázový objekt implementuje. Rozhraní zajišťuje striktní oddělení logické a datové vrstvy.

6.1.2 View - Pohled

Aplikace používá několik pohledů pro interakci s uživatelem. V pohledech jsou komponenty, kterými uživatel ovládá aplikaci. Každý pohled má specifické kontextové menu. V kontextovém menu jsou volby pro ukončení aplikace a nastavení aplikace.

Úvodní pohled zobrazuje seznam zdravotních dokumentací a tlačítko pro vytvoření nové dokumentace. Tyto dokumentace pohledu předá dokumentační řadič. Každá dokumentace ze seznamu lze otevřít jednoduchým kliknutím na název dokumentace. Při kliknutí a přidržení na názvu dokumentace lze dokumentaci vymazat se všemi jejími záznamy. Aplikace se samozřejmě uživatele ještě dotáže, zda chce tuto akci opravdu provést.

Další pohled se zobrazí při kliknutí na tlačítko pro vytvoření nové zdravotní dokumentace z úvodního pohledu. V tomto pohledu uživatel zvolí název, druh a archetyp dokumentace a uloží dokumentaci kliknutím na tlačítko v dolní části pohledu. Název musí být jedinečný. V případě, že název jedinečný není, aplikace vyzve uživatele pro opravu. Druh dokumentace reprezentuje naimplementované druhy zdravotních dokumentací. Pohled dále uživateli nabídne výběr openEHR archetypů, které se nachází na SD kartě mobilního zařízení. Název této složky lze měnit v nastavení aplikace. Výběr archetypů a druhy dokumentací předá pohledu dokumentační řadič. Při uložení archetypu odešle pohled data od uživatele dokumentačnímu řadiči a v případě úspěšného uložení aplikace zobrazí úvodní pohled.

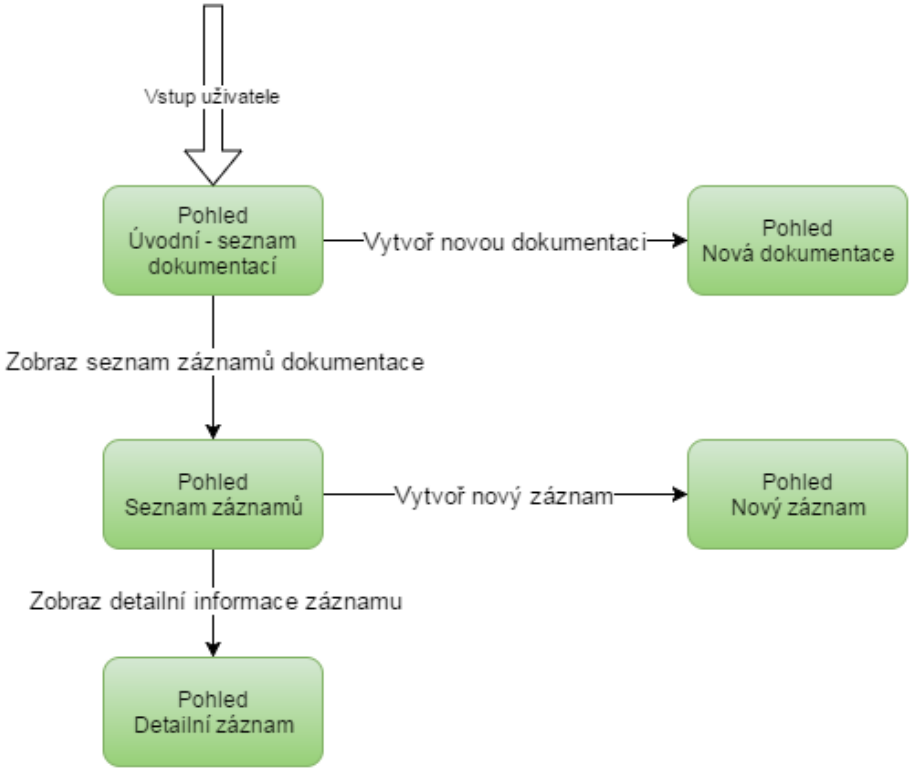
Při kliknutí na jednu ze seznamu dokumentací v úvodním pohledu se vytvoří pohled seznamu záznamů dokumentace. V tomto pohledu je seznam záznamů, které pohledu předal příslušný záznamový řadič a tlačítko pro vytvoření nového záznamu. Libovolný prvek

ze seznamu záznamů lze smazat kliknutím a přidržením na jeho názvu. Aplikace opět uživatele vyzve ke schválení akce. Při kliknutí na jeden ze záznamů aplikace uživateli zobrazí pohled s detailními informacemi o záznamu.

Pohled s detailními informacemi je velice jednoduchý. Zobrazí uživateli výčet informací, které záznam obsahuje. V tomto pohledu není možná žádná interakce uživatele s pohledem. Uživatel se z tohoto pohledu přesune na seznam záznamů kliknutím na tlačítko zpět na svém mobilním zařízení nebo na tlačítko zpět v kontextovém menu.

Kliknutím na tlačítko pro vytvoření nového záznamu v pohledu seznamu záznamů se uživatel přesune na pohled, jehož šablonu vytváří řadič záznamů podle druhu dokumentace. Může být proto libovolný. Na konci pohledu je tlačítko pro uložení záznamu. Kliknutím na toto tlačítko posílá pohled data od uživatele řadiči. Ten je posílá modelu pro uložení do databáze.

Následující obrázek (Obrázek 7) znázorňuje posloupnost pohledů, jak je může uživatel procházet. Uživatel se přesune na další pohled provedením akce, která je popsána na obrázku.



Obrázek 7 Posloupnost pohledů
Zdroj: vlastní

Poslední pohled zobrazuje nastavení aplikace. Zde jsou různé položky, které reprezentují globální atributy aplikace. Do tohoto pohledu se uživatel dostane přes kontextové menu každého pohledu, ve kterém zvolí příslušnou volbu. V nastavení může uživatel měnit např. jméno složky na SD kartě, ve které jsou uložené archetypy. Důležitou skupinou atributů nastavení aplikace je synchronizace dat. V této skupině atributů může uživatel aktivovat nebo deaktivovat synchronizaci dat na mobilním zařízení a se vzdáleným úložištěm dat. Uživatel zde musí také nastavit správnou adresu vzdáleného úložiště dat.

6.1.3 Presenter - Řadič

V aplikaci existují dva druhy řadičů. Řadiče se starají o komunikaci pohledů s datovým modelem. Při vytvoření pohledu se vytvoří příslušný řadič a pohled si uloží jeho referenci. Díky tomu může pohled žádat řadič o potřebná data, nebo vyvolávat příslušné akce.

První druh řadiče se stará o práci s dokumentacemi. Je založen na návrhovém vzoru jedináček. V případě dokumentací není třeba vytvářet více instancí, protože existuje pouze jeden seznam dokumentací. Řadič v sobě uchovává seznam dokumentací, který v případě akce uživatele (vytváření a mazání dokumentace) upravuje. Dokumentační řadič implementuje rozhraní pro pohled seznamu dokumentací a pohled vytvoření nové dokumentace. Každý z těchto pohledů přistupuje k řadiči pomocí jiného rozhraní, což zajišťuje oddělení logické a prezentační vrstvy.

Druhý druh řadiče obstarává práci se záznamy. Pro každou dokumentaci proto musí existovat jeden záznamový řadič. Každý záznamový řadič v sobě uchovává seznam záznamů z určité dokumentace a referenci na tuto dokumentaci. Tento seznam upravuje na základě akcí uživatele. Záznamový řadič implementuje tři rozhraní pro práci s pohledem seznamu záznamů, pohledem detailních informací záznamu a pohledem pro vytvoření nového záznamu. Musí tedy implementovat metody pro získání seznamu záznamů, detailních informací a mazání záznamů.

6.1.4 Moduly

Vytvoření obecné šablony pohledu pro vytváření nových záznamů pro všechny dokumentace není příliš vhodné. V různých dokumentacích může aplikace po uživateli chtít

různá data při vytváření nového záznamu. Pro nový záznam může aplikace také požadovat různá data z mobilního zařízení, na kterém mobilní aplikace běží.

Z tohoto důvodu jsem se rozhodl pro používání modulů pro záznamy z každé dokumentace. Implementací těchto modulů zajistíme vytvoření šablony pro pohled nového záznamu. Také zajistíme získání a zpracování dat od uživatele. Moduly dále zpracují v případě nutnosti externí data z mobilního zařízení. Příkladem může být modul pro sběr záznamů běhání, který by zpracovával data, jako například délka trasy a čas, z běžecké aplikace třetí strany a od uživatele by získal popis terénu a krevní tlak po uražení trasy.

Metody modulu volá třída konkrétní dokumentace, která dědí od třídy obecné dokumentace. Třída modulu implementuje metody pro vytvoření šablon a získání dat z těchto šablon a z mobilního zařízení pro vytváření záznamů z dokumentací, které definuje tato třída konkrétní dokumentace. Výstupní data modulu musí obsahovat openEHR objektový model, který definuje archetyp tohoto modulu.

6.2 Zpracování Sleep as Android dat

V rámci mé bakalářské práce implementuji zdravotní dokumentaci pro ukládání spánkových záznamů. Aplikace by měla být schopná získávat medicínská data z ostatních aplikací na mobilním zařízení, které sběr těchto dat implementují. Pro kompletní sběr spánkových dat je potřeba monitorování uživatele v průběhu spánku. Z tohoto důvodu používám data aplikaci třetí strany Sleep as Android. Tato aplikace je vyvíjena českými vývojáři, se kterými byla původně dohodnutá spolupráce. Bohužel ale od spolupráce odstoupili a já se musel vydat jinou cestou pro získání spánkových dat. Ve své aplikaci využívám soubor, který aplikace Sleep as Android exportuje pro zálohu nebo migraci dat.

Nejprve je nutné monitorovat spánek pomocí této aplikace. Dále je nutné exportovat data do souboru na SD kartě, kde k nim bude mít aplikace pro osobní zdravotní dokumentaci přístup. Z tohoto souboru aplikace získá data o pohybové aktivitě v průběhu spánku, délku spánku, čas probuzení atd. Tato data zpracovávám spolu s daty od uživatele v rámci modulu pro spánek, který jsem implementoval.

Zpracovávat data ze souboru, který vytváří aplikace Sleep as Android jako zálohu nebo migraci dat není ideální způsob získání spánkových dat. Původní záměr byl použít API spánkové aplikace a tím získat spánková data. To však z výše zmíněných důvodů nešlo.

6.3 Databázové rozhraní

Pro ukládání dat jsem vybral databázi Couchbase Lite, která poskytuje kompletní databázové řešení pro mobilní zařízení. Při přístupu k síti dokáže synchronizovat uložená data na mobilních zařízeních se vzdáleným databázovým systémem Couchbase Server. To vytváří ideální podmínky pro vznik a používání osobní zdravotnické dokumentace.

6.3.1 Uložení dokumentací

Při vytvoření nové dokumentace uživatelem pošle pohled zprávu databázovému řadiči. Ten vytvoří novou dokumentaci, uloží jí do seznamu již existujících dokumentací a přepošle instanci databázovému objektu pro uložení do databáze. Databázový objekt vytvoří z instance nové dokumentace JSON dokument, obsahující všechna data dokumentace. Dokument pak uloží do databáze na mobilním zařízení. Databáze po uložení dokumentu vrací řetězec reprezentující jednoznačný identifikátor dokumentu. Tento identifikátor databázový objekt vrací zpět řadiči, který nastaví identifikátor nově vzniklé dokumentaci.

6.3.2 Uložení záznamů

Uložení záznamů funguje podobně jako uložení dokumentace. Pohled zašle zprávu s daty nového záznamu příslušnému řadiči. Řadič získá data nezbytná pro uložení záznamu z konkrétní dokumentace, která je součástí modulu. Tento modul implementuje druh dokumentace, do které záznam patří. Řadič dále získá aktuální datum a čas vytvoření záznamu a odešle data databázovému objektu, který data uloží do databáze. Řadič nakonec informuje pohled o úspěšnosti uložení záznamu do databáze.

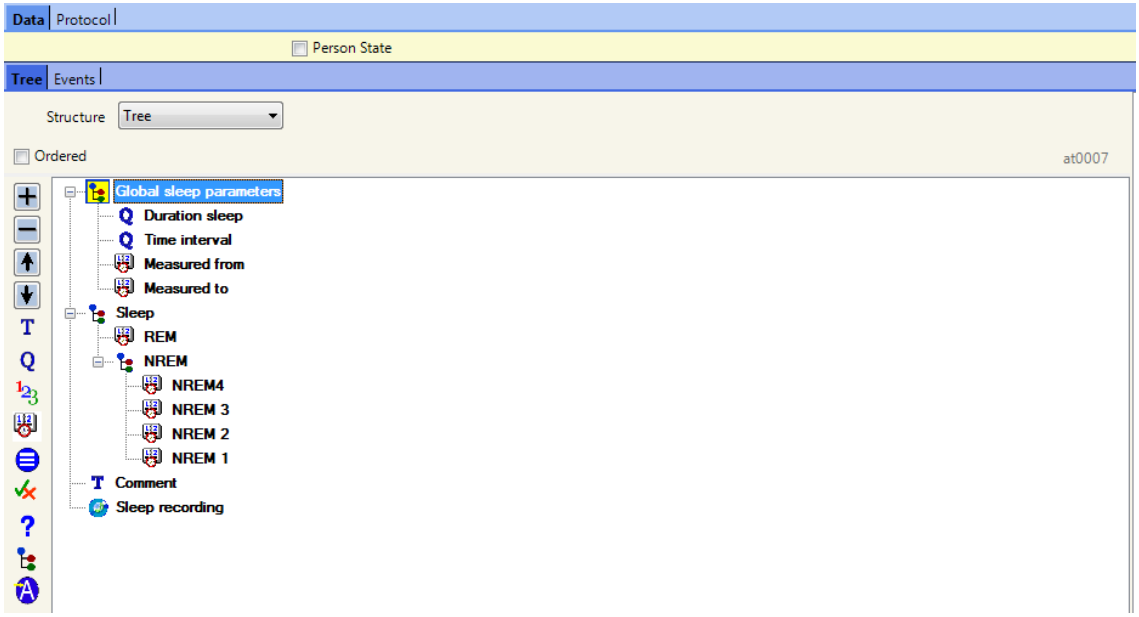
7. SPÁNKOVÝ OPENEHR ARCHETYP

Pro formální popis spánku, dle konceptu openEHR, je třeba použít openEHR archetyp. V současné době neexistuje takový archetyp, který by spánek popisoval. Proto bylo třeba takový archetyp vytvořit. Nadace openEHR vytvořila pro tento účel Archetype editor, což je nástroj pro tvorbu a editaci archetypů v různých jazycích.

Archetyp openEHR referenčního modelu typu OBSERVATION má, mimo jiné, datovou a protokolovou část. Také má část, ve které je hlavička archetypu. V této části jsou různé popisky archetypu. V datové části jsou formálně popsaná spánková data. V protokolové části je formálně popsané informace o sběru těchto dat. Termíny týkající se spánku použité v datové části archetypu mám podložené terminologií SNOMED CT a spánkovou ontologií (viz kapitola Spánková ontologie). Archetyp je napsaný v jazyce pro definici archetypů ADL (Archetype Definition Language).

7.1 Data

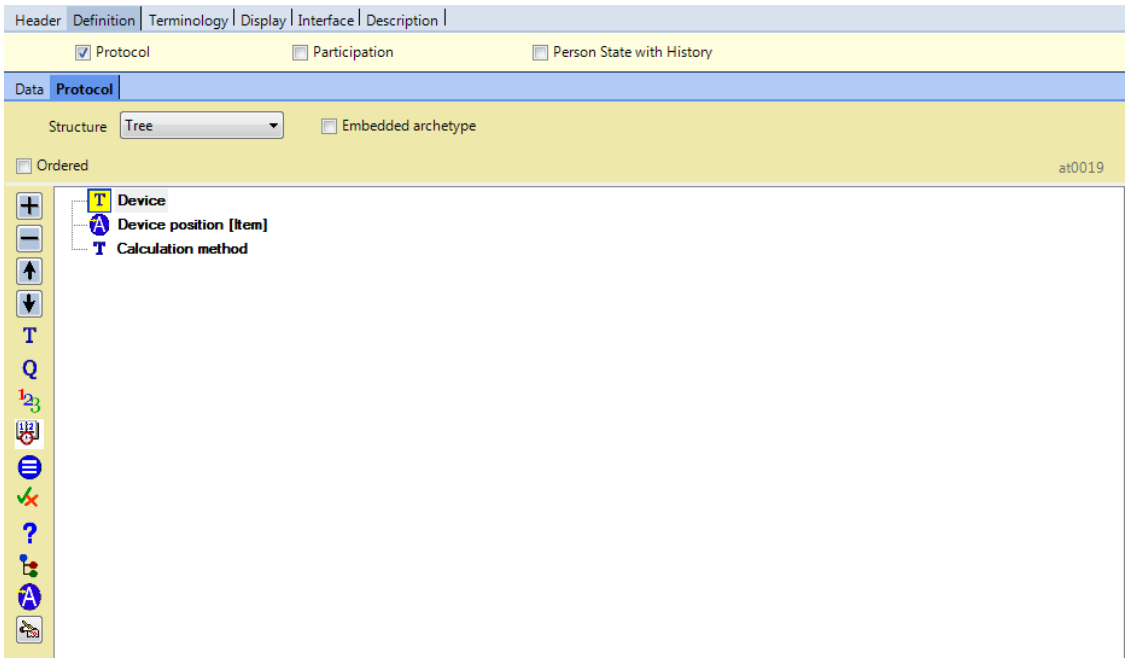
Datová část archetypu se dělí na čtyři hlavní atributy popisující průběh spánku. Prvním atributem jsou globální parametry. V globálních parametrech se určuje hodnota celkové doby spánku, délka časových intervalů, ve kterých se zaznamenává pohybová aktivita uživatele, datum a čas začátku a konce monitorování spánku. Dalším atributem je spánek. Tento atribut zahrnuje spánek REM a NREM. Součástí NREM spánku jsou i čtyři jeho fáze. Třetím atributem je komentář, respektive popis kvality spánku uživatele. Posledním atributem je nahrávání spánku. Tento atribut může obsahovat zvukovou stopu z průběhu spánku nebo například polygrafický záznam spánku (viz Obrázek 8).



Obrázek 8 Datová část spánkového archetypu
Zdroj: vlastní

7.2 Protokol

V protokolové části archetypu jsou tři atributy popisující metadata monitorování spánku. Prvním atributem jsou informace o zařízení, které spánek uživatele monitorovalo. Tento atribut může obsahovat jméno, tovární označení a detailní informace o zařízení. Druhým atributem je poloha monitorovacího zařízení. Poloha zařízení je relevantní informace zvláště v případě monitorování spánku mobilním zařízením. Posledním atributem je kalkulační metoda. Tento atribut definuje například rovnici, která rozhoduje o klasifikaci fáze spánku, ve které se spící právě nachází (viz Obrázek 9).



Obrázek 9 Protokolová část spánkového archetypu
Zdroj: vlastní

7.3 Ontologie

Termíny použité v datové části jsou navázané na světově uznávanou terminologii Snomed-CT. Dále jsou všechny termíny z datové části opatřené vysvětlivkami, které čerpám z kapitoly Spánková ontologie. Tím by měly být veškeré termíny a definice dostatečně podložené. V protokolové části nejsou použité žádné odborné termíny, které by potřebovaly být podložené.

V následující tabulce (viz Tabulka 2) jsou zobrazené termíny navázané na terminologii Snomed International Clinical Terms 2002. První položka je kódová značka používaná v archetypu.

ID	Název	Snomed kód	Popis	Typ
at0000	Sleep	258158006	Detailní informace v průběhu spánku	Cluster
at0004	Duration sleep	309610004	Počet hodin od usnutí do probuzení	Quantity
at0005	Measured from	263571004	Datum a čas začátku měření	Date and time
at0006	Measured from	263572006	Datum a čas konce měření	Data and time
at0009	Time interval	385673002	Časový interval, za který se vzorkuje pohybová aktivita	Quantity
at0017	REM	89129007	Spánková fáze Rapid-Eye-Movement	Time only
at0018	NREM	60984000	Spánková fáze Non-Rapid-Eye-Movement	Cluster
at0010	NREM 1		Spánková fáze Non-Rapid-Eye-Movement 1	Time only
at0011	NREM 2		Spánková fáze Non-Rapid-Eye-Movement 2	Time only
at0012	NREM 3		Spánková fáze Non-Rapid-Eye-Movement 3	Time only
at0013	NREM 4		Spánková fáze Non-Rapid-Eye-	Time only

		Movement 4	
at0014	Comment	Komentář uživatele o kvalitě spánku	Text
at0015	Sleep recording	Zvuková stopa, EEG graf nebo například polygrafický záznam	MultiMedia
Protokolová část			
at0019	Device	Název a detailní informace použitého zařízení	Text
at0020	Device position	Pozice zařízení v průběhu spánku	Slot
at0021	Calculation method	Definuje způsob klasifikace fáze spánku.	Text

Tabulka 2Navázání termínů na terminologii SNOMED-CT

Zdroj: vlastní

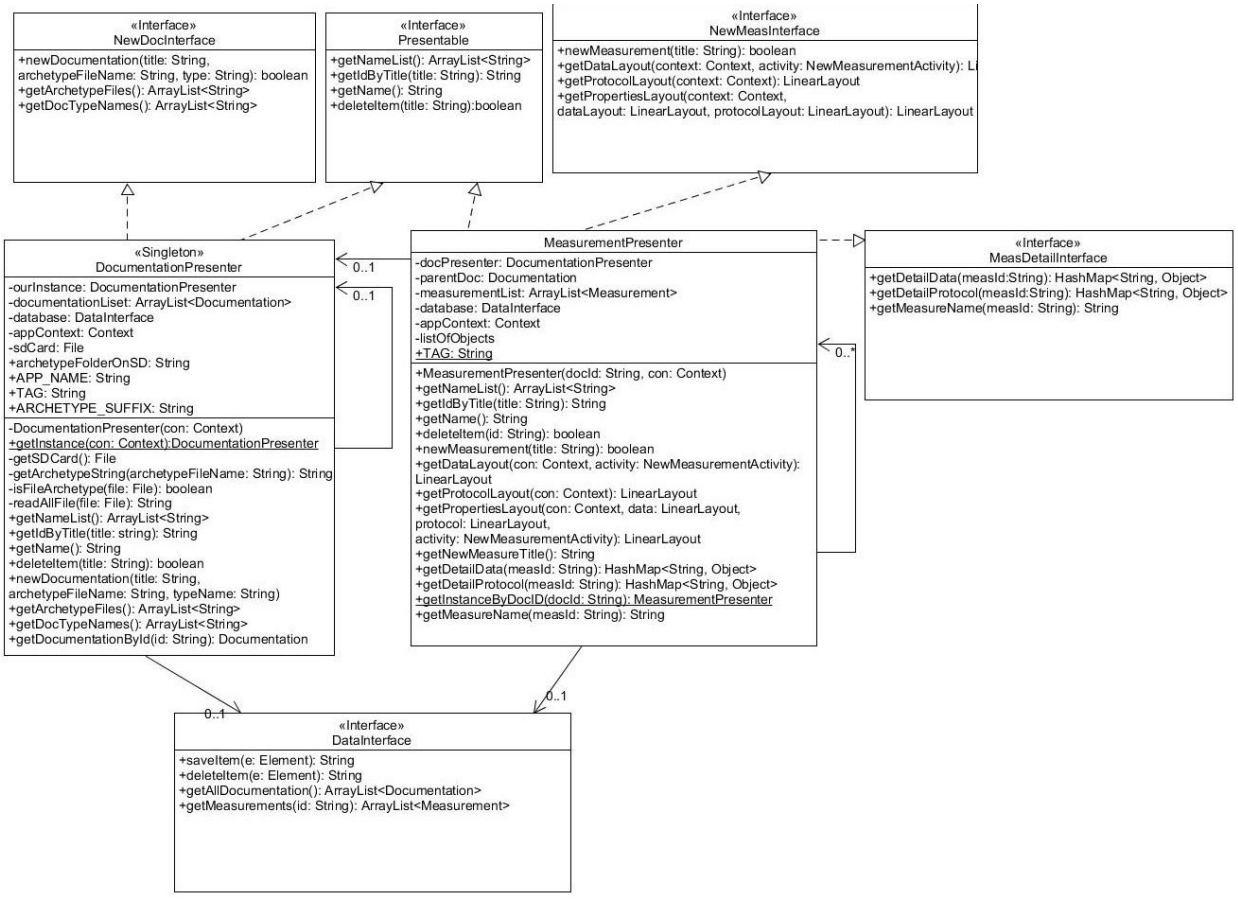
8. IMPLEMENTACE

Celou aplikaci jsem implementoval ve vývojovém prostředí Android studio verze 1.5.1. Dále používám pro vývoj Android SDK 21 (Software Development Kit). Sestavení celé aplikace obstarává nástroj Gradle verze 2.8.

V celé aplikaci využívám pomocné logovací výpisy pro ladění aplikace. V Androidu existuje logovací systém logcat, který sbírá a zobrazuje tyto pomocné výpisy. V aplikaci jsou využity výhody objektového programování, které jazyk Java nabízí.

8.1 Řadiče

Řadiče reprezentují logickou vrstvu aplikace. Jsou implementované jako třídy v jazyce java. Pohledy mohou komunikovat s těmito řadiči pomocí příslušných rozhraní, které tyto řadiče implementují. Řadiče používají pro komunikaci s databázovým objektem databázové rozhraní. UML diagram logické vrstvy je vyznačen na obrázku níže (viz Obrázek 10).



Obrázek 10 UML diagram logické vrstvy
Zdroj: vlastní

8.1.1 Dokumentační řadič – DocumentationPresenter.java

Jak již bylo zmíněno, dokumentační řadič je navržen podle návrhového vzoru jedináček. V celé aplikaci bude existovat pouze jedna instance této třídy. Pro získání reference na tuto jedinou instanci musí pohledy volat tovární metodu, kterou tato třída implementuje.

Třída při inicializaci získá referenci na objekt databázového rozhraní. Dále z databáze načte všechny uložené dokumentace a získá pomocí Android API cestu SD karty pro prozkoumání složky s archetypy.

Třída implementuje rozhraní Presentable. Musí tedy implementovat jeho metody pro získání jmen dokumentací, získání jména aplikace a smazání dokumentace. První metoda slouží k získání jmen dokumentací pro zobrazení seznamu dokumentací v úvodním pohledu aplikace. To zařídí řadič jednoduše, protože seznam dokumentací je jedním z jeho atributů. Další metodou je mazání dokumentace. Pří zavolání této metody řadič zavolá metodu pro odstranění dokumentace z databáze. Při úspěšném odstranění z databáze metoda obstará mazání dokumentace ze seznamu dokumentací a vrátí hodnotu true. Poslední metoda slouží k vrácení řetězce názvu aplikace.

Řadič dále implementuje rozhraní NewDocInterface. V rámci tohoto rozhraní implementuje metody pro vytvoření nové dokumentace, získání jmen souborů s archetypy dokumentací na SD kartě a získání jmen implementovaných dokumentací.

Metoda pro získání jmen souborů s archetypy prohledá složku s archetypy, která se nachází na SD kartě. V případě, že tato složka ještě neexistuje, vytvoří ji. Aplikace při nainstalování počítá s názvem složky pro archetypy „MDC_Archetypes“. Tento název lze měnit v nastavení aplikace. Metoda vrací list s názvy všech souborů archetypů s příponou .adl, které se ve složce nachází.

Metoda, která vrací jména implementovaných dokumentací, vrací list řetězců, do kterého ukládá jména všech tříd, které dědí od třídy Documentation. Toto jméno získají voláním statické metody getTypeName().

Při zavolání metody pro vytvoření nové dokumentace řadič nejprve zkontroluje, zda již název dokumentace, který si uživatel přeje uložit, existuje. V případě, že název dokumentace již existuje, upozorní na to uživatele zprávou a odmítne vytvořit dokumentaci. V opačném případě načte obsah souboru archetypu, vytvoří novou instanci požadovaného

typu dokumentace a zavolá metodu databáze pro uložení. Databáze vrací identifikátor, který řadič přiřadí nové dokumentaci. Nakonec vrací hodnotu true, čímž informuje pohled o úspěšném vytvoření nové dokumentace.

8.1.2 Záznamový řadič – MeasurementPresenter.java

Třída MeasurementPresenter implementuje řadič záznamů. Instance této třídy provádějí akce nad skupinou záznamů jedné dokumentace. Tyto instance v sobě uchovávají referenci na objekt dokumentace, s jejímiž záznamy řadič pracuje. Dále obsahuje list záznamů této dokumentace a objekt databázového rozhraní. Při získání záznamů z databáze se ukládají do objektu dokumentace po celou dobu běhu aplikace. Při vytváření řadiče se testuje, zda jsou záznamy již načtené, nebo se teprve budou muset získat z databáze. To snižuje počet přístupu do databáze, a tím zvyšuje rychlost aplikace.

Záznamové řadiče implementují, stejně jako dokumentační řadič, rozhraní Presentable. Implementuje tedy stejné metody, ale s jiným chováním. První metoda vrací seznam jmen záznamů dokumentace. Druhá metoda maže záznam z databáze a ze svého seznamu záznamů.

Dále implementuje rozhraní NewMeasInterface. Toto rozhraní umožňuje pro přístup k řadiči pohledu pro vytvoření nového záznamu. Řadič musí implementovat metody pro vytvoření nového záznamu a pro vytvoření šablony pro pohled. Šablona pohledu je tvořena částí datovou, protokolovou a částí pro dodatečné informace. Všechny tyto části získává z dané dokumentace voláním příslušných metod. V části dodatečných informací uživatel například určí, který modul se má pro vytvoření šablony použít. Řadič dále implementuje metodu pro získání nadpisu pohledu. Při potvrzení uživatele pro vytvoření nového záznamu zavolá pohled metodu řadiče pro vytvoření nového záznamu. Řadič pouze zavolá příslušné metody modulu k získání dat záznamu a zavolá metodu databázového rozhraní pro uložení záznamu.

Poslední rozhraní, které řadič implementuje pro komunikaci s pohledem detailních informací záznamu, je MeasDetailInterface. Zde se implementují pouze metody pro předání datové a protokolové části dat a informace o názvu záznamu.

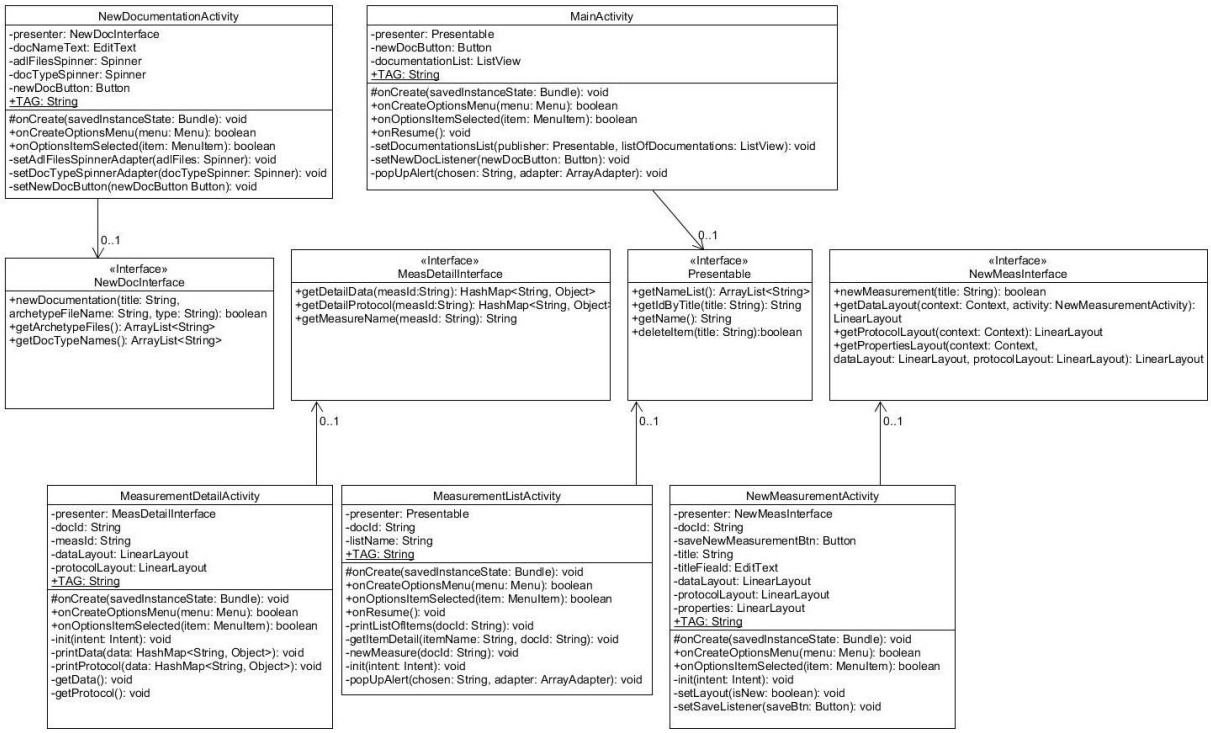
8.2 Aktivity – Pohledy

Aktivity tvoří prezentační vrstvu aplikace. Pohledy aplikace jsou implementovány jako Android aktivity. Aktivita je třída, kterou poskytuje Android API. Tato třída se stará o vytvoření a životní cyklus okna. Po vytvoření okna do něj může aktivita vkládat pohled. V mé aplikaci je pro každý pohled vytvořená aktivita.

Třída Intent slouží k zasílání zpráv aktivitám. Pomocí instance této třídy může jedna aktivita zaslat požadavek na vytvoření jiné aktivitě. Instance mohou dále obsahovat data. Aktivity si tak mohou předávat parametry.

Kromě šablony pohledu nového záznamu jsou všechny šablony pohledů definované v XML souborech. Šablony pohledu nových záznamů jsou generované v modulech. Při vytvoření okna aktivita nastaví pohled. Použije k tomu šablonu z XML souboru a vkládá do ní další komponenty pro interakci s uživatelem.

Všechny aktivity mohou zobrazovat menu možností. V tomto menu může uživatel zvolit možnost ukončení celé aplikace nebo zobrazení nastavení aplikace. Při zvolení volby zobrazení nastavení aplikace se vytvoří nová aktivita, která zobrazuje globální nastavení aplikace. Všechny aktivity, respektive pohledy, komunikují s příslušnými řadiči pomocí rozhraní (viz Obrázek 11).



Obrázek 11 UML diagram prezentační vrstvy
Zdroj: vlastní

8.2.1 Úvodní aktivita – Seznam dokumentací – MainActivity.java

V této aktivitě se zobrazuje seznam všech dokumentací a tlačítko pro vytvoření nové dokumentace. Aktivita nejprve vytvoří rozhraní řadiče pro získání seznamu dokumentací a provádění akcí nad těmito dokumentacemi. Rozhraní vytvoří pomocí tovární metody třídy DocumentationPresenter.

Aktivita dále nastaví událost po kliknutí na tlačítko pro vytvoření nové dokumentace. Po stisknutí tlačítka se vytvoří nová aktivita pro vytvoření nové dokumentace. Současná aktivita se přesune do zásobníku pozastavených aktivit a nově vytvořená aktivita se spustí a přesune na popředí.

Dále od řadiče získá zavoláním příslušné metody seznam dokumentací a zobrazí ho v pohledu. Na zobrazený seznam dokumentací aktivita nastaví odchytávání dvou událostí. První událost je jednoduché kliknutí na prvek ze seznamu dokumentací. Po této události se vytvoří nová aktivita pro zobrazení seznamu záznamů z dokumentace, na kterou se kliklo a současná aktivita se opět pozastaví. Nové aktivitě se předá identifikátor dokumentace, jejíž záznamy bude zobrazovat. Druhá událost je dlouhé kliknutí na prvek ze seznamu. Tato událost slouží k jednoduchému nenávratnému odstranění dokumentace se všemi jejími záznamy. Před definitivním smazáním se zobrazí dialogové okno, které uživatele vyzve k potvrzení či zrušení akce mazání dokumentace. Při smazání dokumentace aktivita zavolá příslušnou metodu řadiče pro provedení změn.

8.2.2 Aktivita nové dokumentace – NewDocumentationActivity.java

V této aktivitě se zobrazuje pohled pro vytvoření nové dokumentace. Aktivita nejprve získá instanci dokumentového řadiče. Pomocí řadiče získá pohled seznam archetypů na SD kartě a seznam implementovaných typů dokumentace. Tyto seznamy tvoří datový model komponent, které umožní uživateli výběr archetypu a typu dokumentace. V případě chybějících archetypů na SD kartě aktivita informuje uživatele pomocí zprávy, skončí a do popředí přejde aktivita seznamu dokumentací.

Dále pohled zobrazuje textové pole pro název dokumentace a tlačítko pro uložení nové dokumentace. Při kliknutí na tlačítko pohled odešle data řadiči pro uložení nové dokumentace. Při duplicitním jméně dokumentace řadič vrátí negativní odpověď a odmítne vytvořit novou dokumentaci. Pohled na to upozorní uživatele. Při úspěšném uložení

dokumentace řadič vrací hodnotu true. V tomto případě pohled končí a úvodní pohled seznamu dokumentací se dostává na popředí.

8.2.3 Aktivita seznamu měření – MeasurementListActivity.java

Tato aktivita zobrazuje seznam existujících záznamů v rámci určité dokumentace. V dolní části pohledu se nachází tlačítko pro vytvoření nového záznamu. Vstupní parametr aktivity je identifikátor dokumentace, jejíž záznamy bude aktivita zobrazovat.

Aktivita nejprve získá objekt řadiče pro záznamy. Při vytváření řadiče předá aktivita identifikátor dokumentace konstruktoru řadiče. Tím zajistí vytvoření řadiče pro správu záznamů správné dokumentace. Zavoláním příslušné metody řadiče získá aktivita seznam záznamů dokumentace. Dále pomocí řadiče získá název seznamu měření a zobrazí ho.

Aktivita zobrazuje tlačítko pro vytvoření nového záznamu. Událost stisknutí tohoto tlačítka se obslouží vytvořením nové aktivity pro vytvoření nového záznamu. Této aktivitě předá identifikátor dokumentace.

Aktivita dále zobrazí seznam záznamů. V tomto seznamu se odchytávají události jednoduchého a dlouhého kliknutí, podobně jako u seznamu dokumentací. Dlouhé kliknutí na položku v seznamu záznamů vyvolá metodu řadiče pro smazání záznamu. Aktivita opět vyzve uživatele pro potvrzení akce prostřednictvím dialogového okna. Jednoduché kliknutí vyvolá vytvoření a zobrazení nové aktivity s pohledem detailního zobrazení záznamu, na který bylo kliknuto. Této aktivitě předá identifikátor dokumentace a identifikátor záznamu, který chce uživatel detailně zobrazit.

8.2.4 Aktivita detailu měření – MeasurementDetailActivity.java

Tato aktivita zobrazuje pohled výčtu informací konkrétního záznamu referenčního modelu typu Observation. Proto v pohledu existuje datová a protokolová část. Vstupními parametry aktivity jsou identifikátor dokumentace a identifikátor záznamu. Pomocí identifikátoru dokumentace získá aktivita řadič příslušných záznamů. Oproti aktivitě seznamu záznamů získá instanci pomocí tovární metody třídy řadiče záznamů. To zařídí předání existujícího řadiče, namísto vytvoření nového a tím šetří místo v paměti. Tento postup je zapříčiněn také tím, že jednotlivé aktivity si nemohou předávat referenci řadiče,

aniž by řadič musel implementovat rozhraní Serializable. Aktivita od řadiče získá název záznamu a vytiskne ho.

Aktivita dále získá datové a protokolové informace záznamu voláním příslušné metody řadiče. Tyto informace jsou v podobě dvou instancí třídy HashMap. Dále se tyto informace vypíšou iteračním způsobem do datové a protokolové části pohledu.

8.2.5 Aktivita nového měření – NewMeasurementActivity.java

Tato aktivita slouží k zobrazení pohledu pro vytvoření nového záznamu referenčního modelu Observation. Pohled zobrazí tři části šablony a tlačítko pro uložení. V první části se zobrazí komponenty pro sběr a zobrazení dodatečných informací. Ve druhé a třetí části se zobrazují komponenty pro sběr a zobrazení datových a protokolových informací. V dolní části pohledu se nachází tlačítko pro uložení dat nového záznamu.

Vstupním parametrem aktivity je identifikátor dokumentace. Pomocí tohoto identifikátoru získá aktivita referenci na řadič záznamů pomocí tovární metody třídy řadiče záznamů stejně jako v aktivitě zobrazení detailu záznamu. Aktivita dále získá zavoláním příslušné metody řadiče nadpis a zobrazí ho.

Dále aktivita získá od řadiče šablonu pro část dodatečných informací, datovou a protokolovou část a zobrazí jí v pohledu.

Po stisknutí tlačítka pro uložení se zavolá metoda řadiče, který data zpracuje. V případě úspěšného zpracování a uložení dat řadič informuje pohled. Ten informuje uživatele a ukončí současnou aktivitu. To vyvolá přesun aktivity seznamu záznamů dokumentace na popředí.

8.2.6 Aktivita nastavení aplikace – SettingsActivity.java

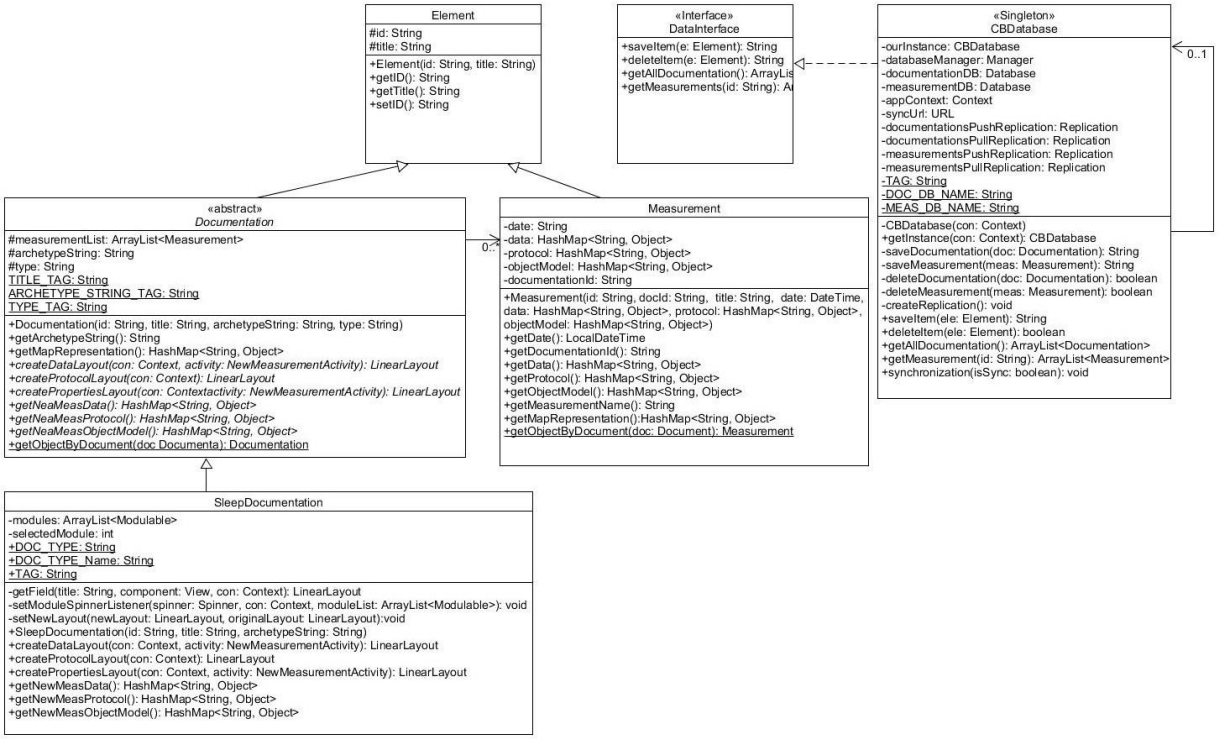
Tato třída dědí od třídy PreferenceActivity, která slouží pro nastavení globálních parametrů aplikace. Tato aktivita obstarává interakci s uživatelem, data i práci s nimi. Nepoužívá tedy žádný řadič, čímž se odlišuje od ostatních aktivit aplikace. To je dáno principem používání Android API pro nastavení aplikace.

Pohled v této aktivitě je vytvořen v XML souboru preferences.xml. Zde je explicitně deklarovaná šablona pro pohled nastavení aplikace. V nastavení lze např. upravit název složky na SD kartě, ve které jsou uložené archetypy.

Dále je v šabloně definovaná kategorie pro synchronizaci dat na mobilním zařízení se vzdáleným úložištěm. První položkou této kategorie je volba, zda má být synchronizace aktivována. V případě změny této volby aktivita upozorní databázový objekt zavoláním příslušné metody, který synchronizaci spustí nebo zastaví. Další položky této kategorie je adresa vzdáleného úložiště dat, se kterým bude aplikace data synchronizovat.

8.3 Model

Datový model reprezentuje vrstvu pro práci s daty. Model je tvořen třídami pro reprezentaci datových objektů a databázového objektu pro práci s daty v databázi na mobilním zařízení. UML diagram datového modelu je na obrázku níže (viz Obrázek 12).



Obrázek 12 UML diagram datové vrstvy
Zdroj: vlastní

8.3.1 Element – Element.java

Element reprezentuje jednoduchou třídu objektů, které mohou být uložené v databázi. Je to základní jednotka datového modelu. Obsahuje atribut identifikátor, který je datového typu řetězec a identifikuje objekt v aplikaci a zároveň v databázi. To je dáno reprezentací identifikátoru dokumentu, ve kterém je element uložen, vytvořeného databází. Pro pořádek a jednoduchou manipulaci tedy aplikace používá tento identifikátor pro instance této třídy, respektive její potomky. Dále element obsahuje textový atribut popisek, který vytváří uživatel pro lidsky zpracovatelnou reprezentaci objektu. Tento atribut musí být jedinečný v rámci elementů stejné třídy.

8.3.2 Dokumentace – Documentation.java

Tato třída definuje objekty, které reprezentují jednotlivé zdravotní dokumentace. Třída dokumentací je abstraktní, protože nikdy nebudeme muset vytvářet instanci obecné dokumentace. Budeme však muset implementovat potomky této třídy včetně abstraktních metod definovaných v této třídě. Potomci budou reprezentovat různé druhy dokumentací. V případě spánkové dokumentace se musí implementovat její třída, která bude dědit od této třídy. Tato třída dále dědí od třídy Element.

Třídy obsahuje atributy seznam záznamů a typ dokumentace. V případě potřeby záznamů této dokumentace se záznamy načtou z databáze a uloží do seznamu záznamů v daném objektu dokumentace. V případě jejich opětovné potřeby se již nenačítají z databáze, ale z tohoto seznamu. Typ dokumentace definuje řetězec, který identifikuje druh dokumentace.

Třída implementuje metody pro získání mapové reprezentace dokumentací a statickou metodu pro získání instance dokumentace správného typu z databázového dokumentu. Metoda získání mapové reprezentace dokumentací vrací instanci třídy HashMap do které vloží veškeré její informace. Statická tovární metoda pro získání instance dokumentace vrací instanci správného typu dokumentace vytvořenou z databázového dokumentu, ze kterého získá veškeré potřebné atributy.

Dále třída definuje abstraktní metody pro vytvoření šablony datové části, protokolové části a části dodatečných informací v pohledu pro vytvoření nového záznamu. Tyto metody budou implementované v potomcích této třídy, respektive v implementacích různých druhů

dokumentací. K těmto třem metodám existují tři abstraktní metody pro sběr dat z těchto šablon. Každá dokumentace bude vytvářet jinou šablonu pro sběr dat od uživatele a bude sbírat jiná data z mobilního zařízení.

8.3.3 Záznam měření – Measurement.java

Tato třída definuje objekty, které reprezentují jednotlivé záznamy ze zdravotních dokumentací. Třídy záznamů dědí od třídy Element. Objekty v sobě obsahují datum, kdy byly vytvořeny. Dále v sobě obsahují instance třídy HashMap pro datovou a protokolovou část záznamu měření openEHR referenčního modelu typu Observation pro zobrazení v detailním pohledu záznamu. Dále v sobě obsahuje openEHR objektový model pro validaci archetypem. Posledním atributem je identifikátor dokumentace, do které záznam patří.

Třída implementuje metodu pro získání jména objektu. Jméno se skládá z popisku a data, kdy byl záznam vytvořen, oddělené pomlčkou. Dále třída implementuje metody pro převedení objektu na mapovou reprezentaci, do které uloží veškeré informace objektu.

Třída dále implementuje statickou metodu pro získání instance této třídy z databázového dokumentu. Z tohoto dokumentu získá atributy mapové reprezentace objektu. A z jednotlivých atributů v mapové reprezentaci vytvoří instanci záznamu, kterou tato metoda vrací.

8.3.4 Spánková dokumentace – SleepDocumentation.java

Spánková dokumentace je příkladem implementace konkrétní dokumentace, která dědí od abstraktní třídy Documentation.java. Tato třída reprezentuje spánkovou dokumentaci, která bude využívat data z aplikace SleepAsAndroid pomocí spánkového modulu.

Při vytvoření instance této třídy se vytvoří seznam dostupných spánkových modulů, které implementují metody pro práci se spánkovými daty.

Třída implementuje metody pro získání šablony pro pohled vytvoření nového spánkového záznamu. Šablonu tvoří část protokolová, datová a část dodatečných informací, musí tedy implementovat tři metody pro získání šablony. Metody pro získání protokolové

a datové části pouze získají tyto části z příslušného modulu a vrátí je řadiči. Metodu pro získání části dodatečných informací jako např. informace, který modul se má použít, implementuje dokumentace sama.

V metodě pro získání dodatečných informací objekt třídy vytvoří komponentu, pomocí které uživatel zvolí modul pro práci se spánkovými daty. Na tuto komponentu je nastavené odchytávání události změny modulu. Pří změně modulu se znovu zavolají metody pro získání protokolové a datové části šablony a vykreslí se do pohledu.

Třída dále implementuje metody pro získání dat ze šablony, kterou vytváří výše popsané metody a k získání objektového modelu spánkových dat. K získání těchto dat metody zavolají příslušné metody právě vybraného modulu.

8.3.5 Databáze – CBDatabase.java

Databázový objekt používá pro uložení dat knihovnu Couchbase Lite, proto musí importovat potřebné třídy této databázové knihovny. Je to dokumentově orientovaná databáze. Do databáze tedy ukládá dokumenty. Dokumenty obsahují datové objekty reprezentované JSON notací. Pro práci řadičů s databází je vytvořeno databázové rozhraní definující metody pro načítání, ukládání a mazání dokumentů v databázi.

Třída databáze je implementovaná podle návrhového vzoru jediáček. Tím umožníme přístup k datům současně maximálně jednomu objektu, což zajistí konzistenci dat. Implementuje tedy opět tovární metodu pro získání jediné instance. Tato instance je uložena ve statické proměnné v databázové třídě. V případě, že instance ještě neexistuje, tovární metoda jí v případě nutnosti vytvoří.

Při vytváření databázového objektu se musí inicializovat objekt Manager, který spravuje databáze a řídí přístup k těmto databázím. Pomocí volání tovární metody Manageru získáme instanci objektu databází. Tímto způsobem získáme objekt databáze dokumentací a databáze záznamů z dokumentací. Veškeré tyto akce jsou ošetřeny odchytáváním výjimek.

Databázové rozhraní pro práce řadičů s databází definuje metody pro uložení a mazání elementů. To, o jaký element se jedná (dokumentace nebo záznam), zjišťuje až databázový objekt. Všechny řadiče tak používají jedno databázové rozhraní. Nemusí se tak

starat jakou metodu volat. Po zavolání metody databázového objektu pro mazání nebo uložení elementu zjistí objekt, zda se jedná o dokumentaci, nebo záznam a podle toho zavolá příslušné metody.

Pro uložení dokumentace předává řadič databázovému objektu instanci dokumentace, která má být uložena. Databázový objekt nejprve vytvoří v databázi dokumentací nový dokument a získá jeho identifikátor. Dále získá mapovou reprezentaci dokumentace voláním její metody. Tato metoda vrátí instanci třídy Map, která reprezentuje objekt dokumentace. Tato instance je převeditelná na JSON notaci, kterou databázový Couchbase Lite dokument vyžaduje pro uložení dat. Mapovou reprezentaci databázový objekt vloží do nově vytvořeného dokumentu. Metoda nakonec vrací řadiči zpět identifikátor dokumentu.

Metoda pro uložení dokumentačního záznamu funguje principiálně stejně. Používá však pro uložení dokumentu záznamovou dokumentaci místo dokumentační.

Pro mazání dokumentace z databáze předá řadič databázovému objektu instanci dokumentace. Databázový objekt zjistí, že se jedná o databázi a zavolá příslušnou metodu. Metoda pro mazání dokumentace vrací hodnotu true, když mazání dopadlo úspěšně. V opačném případě vrací hodnotu false. Metoda nejprve získá správný objekt dokumentu z dokumentační databáze pomocí identifikátoru dokumentace, kterou objektu předal řadič. Nad tímto dokumentem zavolá metodu mazání, která vrací hodnotu true, nebo false podle úspěchu metody. Tuto hodnotu metoda vrátí řadiči. Mazání záznamu dokumentace funguje opět stejně s výjimkou použité databáze.

Databázový objekt dále poskytuje prostřednictvím databázového rozhraní metodu pro získání seznamu všech dokumentací. Tato metoda nejprve vytvoří seznam dokumentací, do kterého bude vkládat dokumentace. Dále metoda vytvoří dotaz, který získá veškeré dokumenty z databáze. Dotaz spustí a získá objekt obsahující výsledek dotazu. Tento objekt poté iteračně prochází a vyjímá z něj jednotlivé dokumenty. Z těchto dokumentu získá dokumentace, které vkládá do seznamu dokumentací. Nakonec metoda tento seznam vrací zpět řadiči.

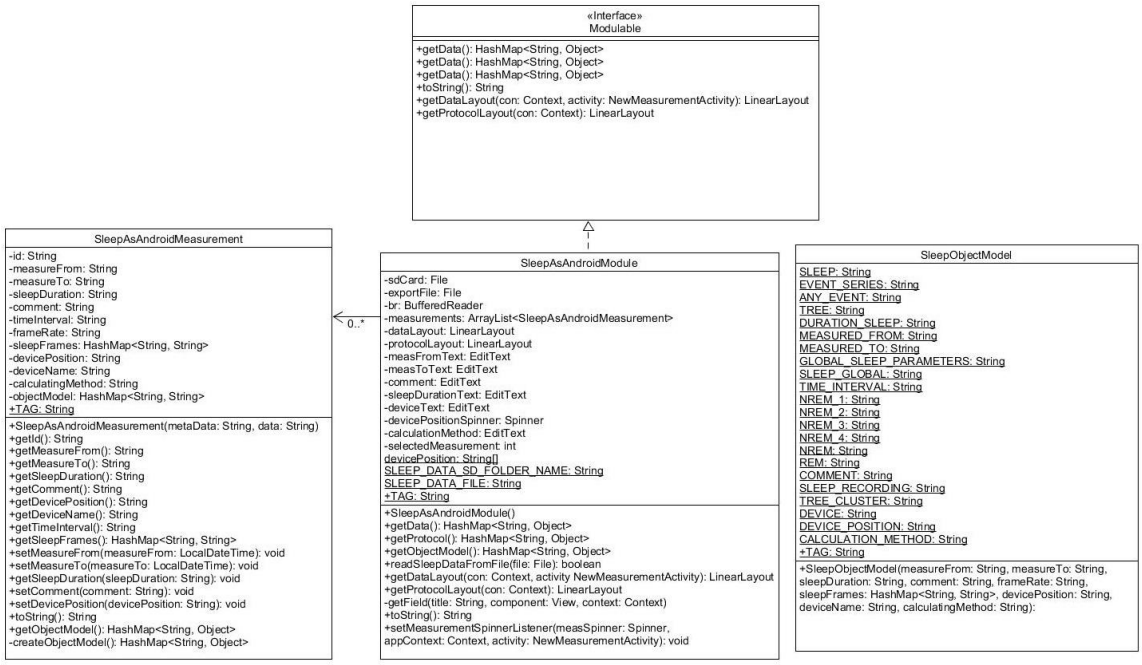
Podobným způsobem funguje metoda pro získání všech záznamů nějaké dokumentace. Metodě musí řadič předat identifikátor dokumentace, ze které chce získat všechny záznamy. Metoda získá ze záznamové databáze všechny záznamy dokumentace s tímto identifikátorem.

8.3.6 Synchronizace

Synchronizace dat aplikace uložených na mobilním zařízení pomocí databázového objektu (viz kapitola Databáze – CBDatabase.java) provádí knihovna Couchbase Lite. Při vytváření databázového objektu se konstruktor pokusí pomocí metod knihovny vytvořit objekty pro obousměrnou replikaci dat z dokumentační a záznamové databáze. Tento postup opakuje databázový objekt i v případě změny adresy vzdáleného úložiště dat v nastavení aplikace. V případě, že je v nastavení aplikace zaškrtnutá položka pro aktivaci Synchronizace, spustí databázový objekt synchronizaci zavoláním příslušných metod objektů pro replikaci dat. Když je adresa vzdáleného úložiště dat neplatná, aplikace na to uživatele upozorní a synchronizaci nespustí.

8.4 Spánkový modul

Spánkový modul je implementovaný jako samostatný Java balíček. Slouží k vytvoření šablony pohledu nového záznamu a zpracování dat s použitím exportovaného souboru aplikace SleepAsAndroid. V balíčku je třída SleepAsAndroidModul, která slouží jako vstupní bod modulu. Dále balíček obsahuje třídu SleepObjectModel pro reprezentaci dat a SleepAsAndroidMeasurement pro extrakci užitečných dat ze souboru, který exportuje aplikace SleepAsAndroid. Implementace spánkového modulu je zobrazena na následujícím UML diagramu (viz Obrázek 13).



Obrázek 13 UML diagram spánkového modulu
Zdroj: vlastní

8.4.1 Vstupní bod modulu – SleepAsAndroidModul.java

Tuto třídu používá záznamový řadič k získání šablony pro pohled nového záznamu a pro následné zpracování dat. Pro komunikaci s řadičem implementuje rozhraní Modulable. Musí implementovat jeho metody pro získání protokolové a datové šablony pohledu nového spánkového záznamu, pro následné získání těchto dat od uživatele a získání objektového modelu s daty vytvořeného podle spánkového archetypu.

Při vytvoření instance této třídy vytvoří seznam SleepAsAndroid měření. Ten naplní jednotlivými záznamy ze souboru, který aplikace SleepAsAndroid exportovala. V případě, že tento soubor neexistuje, nechá seznam prázdný. Záznamy tohoto seznamu jsou instancemi třídy SleepAsAndroidMeasurement, které vytváří objekt pří procházení a analyzování souboru.

Metoda pro tvorbu datové části šablony zobrazuje komponenty pro sběr dat od uživatele, nebo zobrazení dat uživateli. Vytvoří komponentu pro výběr, které SleepAsAndroid měření chce uživatel uložit do dokumentace. Dále zobrazuje datum a čas začátku a konce měření. Tento datum a čas získá modul z exportovaného souboru aplikace SleepAsAndroid a je v pohledu pro vytvoření nového záznamu neměnný. Dále zobrazuje komentář uživatele k měření. Text tohoto komentáře je opět získaný z exportovaného souboru aplikace SleepAsAndroid, ale lze ho změnit.

Metoda pro tvorbu protokolové části vytváří neměnitelné textové pole s informacemi o mobilním zařízení, na kterém aplikace právě běží. Dále vytvoří komponentu, která uživateli umožní výběr pozice mobilního zařízení v proběhu spánku.

Metody pro získání dat z těchto částí šablony jsou jednoduché. Získají pouze hodnoty z komponent, které uživatel získal a uloží je s příslušnými jméno do instance třídy HashMap. Metodu pro získání objektového modelu získá voláním příslušné metody objektu právě vybraného měření.

8.4.2 Objektový model – SleepObjectModel.java

Tato třída dědí od třídy HashMap. Při vytvoření instance této třídy se vytvoří objektový model spánkových dat uložený pod stejnými identifikátory, které jsou definované ve spánkovém archetypu. Pomocí archetypu lze tato data validovat.

8.4.3 Data ze SleepAsAndroid – SleepAsAndroidMeasurement.java

Tato třída reprezentuje jednotlivá měření, která jsou uložená v exportovaném souboru aplikace SleepAsAnddroid. Při vytváření třídy se předají konstruktoru řádky souboru, které se analyzují a získají se z nich užitečné atributy měření. Třída implementuje metodu, která vrací objektový model dat, které obsahuje. Objektový model vytvoří inicializováním objektu SleepObjectModel, kterému při vytvoření předá potřebná data.

9. FUNKČNOST ŘEŠENÍ

S validními vstupními daty funguje aplikace tak, jak by uživatel předpokládal. S nevalidními vstupy aplikace počítá a ošetřuje je. Dokáže vytvářet nové dokumentace a současně je ukládat do databáze. Při vytvoření nové dokumentace se do databáze ukládá její jméno, typ a archetyp, který bude popisovat záznamy v rámci této dokumentace. Aplikace zamezuje vytvoření dokumentace se stejným jménem. Dále aplikace umožňuje mazání dokumentací pomocí dlouhého kliknutí na název dokumentace. Při dostatečném množství volné paměti určené pro databázi je ukládání dat spolehlivé. Při nedostatku volné paměti pro databázi na to aplikace uživatele upozorní a odmítne ukládat další data. Po opětovném spuštění se veškerá data znovu načtou. Vytváření záznamů funguje pomocí modulu pro sběr spánkových dat z aplikace SleepAsAndroid a od uživatele. Uložená data pomocí tohoto modulu si může uživatel kdykoli prohlédnout v detailním pohledu záznamu kliknutím na záznam. Mazání záznamů funguje stejně jako v případě dokumentací.

9.1 Diskuse výsledků

Mobilní aplikace, které pracují s medicínskými daty, existují, nicméně tato data využívají jen pro své účely. Podobné mobilní aplikace zdravotní dokumentace dosud neexistují. Z tohoto důvodu není možné žádného porovnání výsledku.

Dle mého názoru je aplikace dobrý prototyp pro vývoj aplikace osobní zdravotní dokumentace, která bude sloužit pro sběr medicínských dat a pro informování uživatele o jeho zdravotním stavu. Tyto aplikace budou využívat centrální zdravotní dokumentaci, do které bude aplikace vkládat nasbíraná data a zároveň z ní bude používat data, které se k uživateli vztahují.

Relativní nedostatek této aplikace je získávání dat ze souboru, který musí aplikace SleepAsAndroid exportovat. Je to dáno odstoupením vývojářů aplikace SleepAsAndroid od spolupráce, respektive nemožnost použití API této aplikace. Proto jsem zvolil tento alternativní způsob získání spánkových dat z této aplikace.

Díky této bakalářské práci jsem získal velký přehled o implementaci informačních a komunikačních technologií ve zdravotnictví nejen na úrovni konceptu openEHR. Dále jsem získal zkušenosti s vývojem aplikací pro mobilní platformu Android.

9.2 Omezení a hardwarové nároky

Použití openEHR knihoven pro analyzování archetypu v aplikace není možné. Tyto knihovny jsou sice napsané v jazyce Java, který Android podporuje, nicméně knihovny používají funkce, které aplikační rozhraní Android nepodporuje. I přes velkou snahu přepsat tyto rozsáhlé knihovny se jejich zprovoznění nepodařilo. Validace objektového modelu dat by měla v budoucnu probíhat na straně serveru, což je teoreticky lepší řešení. Server by při synchronizaci záznamů spouštěl procedury pro validaci záznamů pomocí archetypů uložených na serveru. Aplikaci je dále vyvíjena pro mobilní zařízení s minimálně Android API 9, což je zapříčiněno Couchbase Lite knihovnami, které toto minimální API vyžadují. Proto je aplikaci možné používat s verzí Android 2.3 a vyšší.

9.3 Testování

Tato kapitola se zaměřuje na testování funkčnosti aplikace. Celá aplikace je testována zejména pomocí předem sestavených scénářů. Tyto scénáře zahrnují běžné i potenciálně nebezpečné chování uživatele, který aplikaci využívá. Testování probíhalo na mém vlastním telefonu Sony Xperia U s Android API 9 a na emulátoru mobilního zařízení s Android API 21, který je součástí Android Studia. Aplikace ošetřuje veškeré potenciálně nebezpečné uživatelské vstupy.

Je ošetřené vytváření duplicitních dokumentací a záznamů. Dále jsou ošetřené nevyplněná textová pole jako např. název dokumentace. V obou případech aplikace uživatele upozorní. Aplikace vyžaduje potvrzení všech destruktivních operací.

Synchronizace dat na mobilním zařízení se vzdáleným serverem je testováno pouze na lokálním zařízení. Zde používám Couchbase Server Couchbase Sync Gateway a emulátor. Couchbase Server i Sync Gateway musí být nainstalován a správně nakonfigurován. Pro správnou konfiguraci doporučuji použít oficiální dokumentaci. Poté stačí v nastavení aplikace zadat správnou adresu serveru a zaškrtnout volbu synchronizace. V případě úspěšného synchronizování aplikace se serverem o tom aplikace uživatele informuje.

Dále aplikace obsahuje jednotkové testy, které testují potenciálně nebezpečné části kódu. Tyto testy automaticky testují např. spustitelnost aplikace po jejím sestavení, správnost vytvoření objektového modelu záznamů z aplikace Sleep As Android, který aplikace vytváří a správnost vytvoření mapové reprezentace záznamů a dokumentací pro uložení do databáze.

10. ZÁVĚR

V rámci této práce se podařilo vytvořit v praxi použitelný openEHR archetyp, který formálně popisuje spánková medicínská data. Tento archetyp je napsaný v jazyce pro definici archetypů a je dostatečně podložený řízenou ontologií a terminologií. Vlastní řešení bylo zvoleno z důvodu absence podobného archetypu ve veřejných repositářích CKM openEHR. Je možné ho použít pro formální popis atributů spánku při jeho monitorování.

Dále se podařilo vytvořit funkční prototyp aplikace pro mobilní zařízení s operačním systémem Android, která sbírá medicínská data a ukládá je do osobní zdravotní dokumentace. Uživatel si může vést svou zdravotní dokumentaci a sledovat díky ní svůj zdravotní stav. Aplikace je dále schopná data synchronizovat s daty na vzdáleném úložišti, čímž umožňuje jednoduché budoucí použití v rámci osobní zdravotní dokumentace vyvíjené pod skupinou Medical Informatics na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni.

Aplikace implementuje případ sběru spánkových dat. Je schopná sbírat data částečně z aplikace třetí strany SleepAsAndroid a částečně od uživatele. Dokáže vytvářet nové dokumentace a ukládat do nich záznamy. Tyto záznamy si může uživatel kdykoliv detailně prohlédnout a může je kdykoliv smazat a vytvořit znova.

Aplikace je připravená na budoucí rozšíření. Přidat novou dokumentaci znamená implementování jedné třídu v jazyce Java, která dědí od třídy obecné dokumentace. Přidání funkce pro sběr dat od uživatele nebo z dalších aplikací lze provést implementací modulu, který bude vytvářet šablonu pro sběr dat od uživatele a bude získávat data z těchto aplikací. Lze například implementovat modul, který bude sbírat data z aplikace, která monitoruje průběh běhání uživatele.

Budoucí vývoj by se měl zaměřit na vyřešení komunikace mezi aplikací a osobní zdravotní dokumentací. To lze obstarat komunikací aplikace s Couchbase serverem, který bude přímo napojený na zdravotní dokumentaci. Součástí komunikace by měla být bezpečná autorizace uživatele mobilní aplikace. Na straně serveru je nutné vytvořit validaci objektového modelu EHR záznamů vytvářených uživateli. Další vývoj aplikace by se týkal rozšíření o další podporované domény, například vytvoření modulu pro sběr dat ze sport-trackerů. V neposlední řadě by si aplikace zasloužila hezčí grafické uživatelské rozhraní.

PŘEHLED ZKRATEK

ADL	Archetype Definition Language
ANSI	American National Standards Institute
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CCD	Continuity of Care Document
CCOW	Clinical Context Object Workgroup
CDA	Clinical Document Architecture
CEN	European Committee for Standardization
EEG	Elektroencefalogram
EHR	Electronic Health Record
EMR	Electronic Medical Record
EU	Evropská unie
HL7	Health Level Seven
HTTP	HyperText Transfer Protocol
IHTSDO	The International Health Terminology Standards Development Organization
Inc.	Incorporated
IoT	Internet of Things
IZIP	český projekt Elektronické zdravotní knížky
Java SE	Java Standard Edition
JDK	Java Development Kit
JSON	JavaScript Object Notation
K-komplex	Potenciál se třemi fázemi následovaný vřetenovitou aktivitou s frekvencí 13 Hz
MVC	Model-View-Controler
MVP	Model-View-Presenter
NČLP	Národní Číselník Laboratorních Položek
NoSQL	Nerelační databáze
NREM	Non Rapid Eye Movement
NZIS	Národní Zdravotnický Informační Systém
OSI	Open Systems Interconnection
PGO spikes	Ponto-Genikulo-Okcipitální hroty
PHR	Personal Health Record
QR kód	Quick Response kód
REM	Rapid Eye Movement
RESTful API	Representation State Transfer Application Programming Interface
RIM	Reference Information Model
SD karta	Secure Digital karta
SDK	Software Development Kit
SPL	Structured Product Labeling
SQL	Structured Query Language
SŘBD	Systém řízení báze dat
XML	Extensible Markup Language
δ-spánek	Stádium spánku, ve kterém dominují z více než 50 % pomalé vlny delta

LITERATURA

1. **Iakovidis, Ilias, Wilson, Petra a Healy, Jean.** *E-Health.* Amsterdam : IOS Press, 2004. ISBN 1-58603-448-0.

2. What is an electronic health record. *HealthIT.gov.* [Online] Office of the National Coordinator for Health Information Technology, 16. Březen 2013. [Citace: 5. Duben 2016.] https://www.healthit.gov/providers-professionals/faqs/what-electronic-health-record-ehr.

3. **Garrett, Peter a Seidman, Joshua.** EMR vs EHR – What is the Difference? *HealthITBUZZ.gov.* [Online] Office of the National Coordinator for Health Information Technology, 4. Leden 2011. [Citace: 6. Duben 2016.] https://www.healthit.gov/buzz-blog/electronic-health-and-medical-records/emr-vs-ehr-difference/.

4. What is a personal health record? *HealthIT.gov.* [Online] Office of the National Coordinator for Health Information Technology, 2. Květen 2013. [Citace: 4. Duben 2016.] https://www.healthit.gov/providers-professionals/faqs/what-personal-health-record.

5. **Svoboda, Filip.** eHealth v EU. *eZDRAV.cz.* [Online] Filip Svoboda, 15. Srpen 2012. [Citace: 12. Duben 2016.] http://www.ezdrav.cz/ehealth-v-eu/. ISSN 1805-7535.

6. —. eHealth v ČR. *eZDRAV.cz.* [Online] Filip Svoboda, 15. Srpen 2012. [Citace: 6. Duben 2016.] http://www.ezdrav.cz/ehealth-v-cr/. ISSN 1805-7535.

7. About HL7. *HL7 International.* [Online] Health Level Seven International. [Citace: 9. Duben 2016.] http://www.hl7.org/about/index.cfm?ref=nav.

8. Introduction to HL7 Standards. *HL7 International.* [Online] Health Level Seven International. [Citace: 9. Duben 2016.] http://www.hl7.org/implement/standards/index.cfm?ref=nav.

9. HL7 Standards - Section 1: Primary Standards. *HL7 International.* [Online] Health Level Seven International. [Citace: 9. Duben 2016.] http://www.hl7.org/implement/standards/product_section.cfm?section=1&ref=nav.

10. HL7 Version 2 Product Suite. *HL7 International.* [Online] Health Level Seven International. [Citace: 9. Duben 2016.] http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185.

11. HL7 Version 3 Product Suite. *HL7 International.* [Online] Health Level Seven International. [Citace: 10. Duben 2016.] https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186.

12. The CEN/ISO EN13606 standard. *EN 13606 ASSOCIATION.* [Online] The EN 13606 Association. [Citace: 12. Duben 2016.] http://www.en13606.org/the-ceniso-en13606-standard.

13. CEN/ISO EN13606 Reference Model. *EN 13606 ASSOCIATION.* [Online] The EN 13606 Association. [Citace: 12. Duben 2016.] http://www.en13606.org/the-ceniso-en13606-standard/reference-model.

14. CEN/ISO EN13606 Archetype Model. *EN 13606 ASSOCIATION.* [Online] The EN 13606 Association. [Citace: 12. Duben 2016.] http://www.en13606.org/the-ceniso-en13606-standard/archetype-model.

15. What is openEHR? *openEHR.* [Online] openEHR Foundation. [Citace: 10. Duben 2016.] http://www.openehr.org/what_is_openehr.

16. Clinical Models Program. *openEHR.* [Online] openEHR Foundation. [Citace: 10. Duben 2016.] http://www.openehr.org/programs/clinicalmodels/.

17. What is SNOMED CT? *ihtsdo.org.* [Online] International Health Terminology Standards Development Organisation. [Citace: 9. Duben 2016.] http://www.ihtsdo.org/snomed-ct/what-is-snomed-ct.

18. How Does SNOMED CT Work? *ihtsdo.org.* [Online] International Health Terminology Standards Development Organisation. [Citace: 9. Duben 2016.] http://www.ihtsdo.org/snomed-ct/what-is-snomed-ct/how-does-snomed-ct-work.

19. Why SNOMED CT? *ihtsdo.org.* [Online] International Health Terminology Standards Development Organisation. [Citace: 9. Duben 2016.] http://www.ihtsdo.org/snomed-ct/why-should-i-get-snomed-ct.

20. About LOINC. *LOINC.* [Online] Regenstrief Institute, Inc., 17. Červen 2015. [Citace: 11. Duben 2016.] https://loinc.org/background.

21. DASTA a projekty e-Health. *DASTA .* [Online] 2012. [Citace: 29. Duben 2016.] http://www.dastacr.cz/info.html.

22. **Lacko, Ľuboslav.** *Vývoj aplikací pro Android.* Brno : Computer Press, 2015. ISBN 978-80-251-4347-6.

23. Smartphone OS Market Share, 2015 Q2. *IDC.* [Online] IDC Research, Inc., Srpen 2015. [Citace: 25. Duben 2015.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

24. **Jiří Vávrů, Miroslav Ujbányai.** *Programujeme pro android 2., rozšířené vydání.* Praha : Grada Publishing a.s., 2013.

25. **Friesen, Jeff.** *Learn Java for Android Development.* New York : Springer Science+Business Media, 2014. ISBN-13 978-1-4302-6455-2.

26. **Grant, Allen.** *Android 4 Průvodce programováním mobilních aplikací.* Brno : Computer Press, 2013. ISBN 978-80-251-3782.

27. Android Studio Overview. *Android Developers.* [Online] Google. [Citace: 4. 4 2016.] http://developer.android.com/tools/studio/index.html.

28. **Leavitt, Neal.** Will NoSQL Databases Live UP to Their Promise? *Leavitt Communication.* [Online] Leavitt Communications, Inc., 2010. [Citace: 11. Duben 2016.] http://leavcom.com/pdf/NoSQL.pdf.

29. Introduction. *Couchbase.* [Online] COUCHBASE. [Citace: 13. Dube 2016.] http://developer.couchbase.com/documentation/server/4.1/introduction/intro.html.

30. Couchbase Mobile. *Couchbase.* [Online] COUCHBASE. [Citace: 13. Duben 2016.] http://developer.couchbase.com/documentation/mobile/1.2/get-started/couchbase-mobile-overview/index.html.

31. Elasticsearch. *elastic.* [Online] Elasticsearch. [Citace: 12. Duben 2016.] https://www.elastic.co/products/elasticsearch.

32. **Rokyta, Richard a kolektiv.** *Fyziologie.* Praha : ISV nakladatelství, 2000. ISBN 80-85866-45-5.

33. **Borzová, Claudia, a další.** *Nespavost a jiné poruchy spánku.* Praha : Grada Publishing a.s., 2009. ISBN 978-80-247-2978-7.

34. **Heřman, Petr.** *Biosignály z pohledu biofyziky. 1. vydání.* Praha : Důlos, 2006. ISBN 80-902899-7-5.

35. **Mourek, Jindřich.** *Fyziologie.* Praha : Grada Publishing a.s., 2005. ISBN 80-247-1190-7.

36. Sleep as Android Unlock. *Google play.* [Online] Google. [Citace: 10. Duben 2016.] https://play.google.com/store/apps/details?id=com.urbandroid.sleep.full.key&hl=cs.

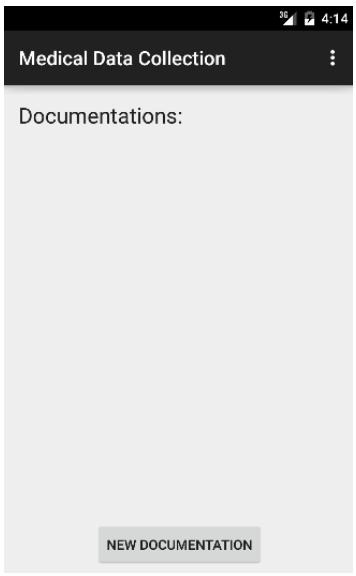
37. SleepBot - Sleep Cycle Alarm. *Google Play.* [Online] Google. [Citace: 10. Duben 2016.] https://play.google.com/store/apps/details?id=com.lslk.sleepbot.

38. How it works. *Sleep Cycle.* [Online] Northcube AB. [Citace: 10. Duben 2016.] http://www.sleepcycle.com/howitworks.html.

PŘÍLOHY

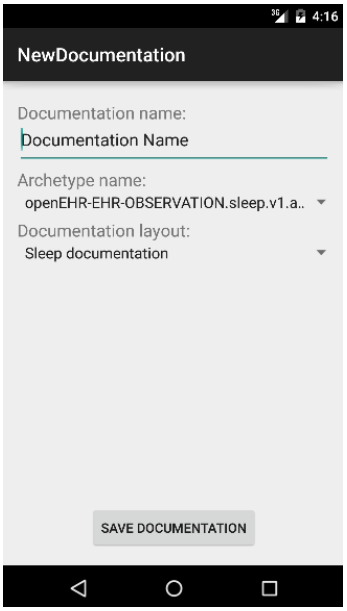
Uživatelská příručka

Aplikace se ovládá velice jednoduše a intuitivně. Je možné jí využívat na mobilních zařízeních s Android 2.3 a vyšší. Při spuštění aplikace uvidí uživatel následující pohled (viz Obrázek 14). V tomto pohledu se zobrazuje seznam existujících dokumentací. Uživatel zde může kliknutím na tlačítko ve spodní části obrazovky vytvořit novou dokumentaci.



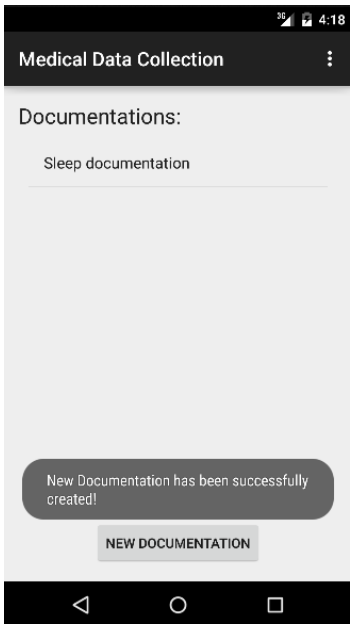
Obrázek 14 Úvodní pohled - seznam dokumentací (prázdný)
Zdroj: vlastní

Po kliknutí na tlačítko se uživateli zobrazí následující pohled (viz Obrázek 15). V tomto pohledu může uživatel vyplnit název nové dokumentace, zvolit jeden z archetypů, který je uložený na SD kartě v příslušné složce a typ dokumentace. Název složky s archetypy lze měnit v nastavení aplikace. V tuto chvíli je na výběr pouze spánkový typ dokumentace. Po validním vyplnění všech položek a kliknutí na tlačítko v dolní části pohledu, se nová dokumentace uloží do databáze, vloží se do seznamu dokumentací v úvodním pohledu a upozorní uživatele na tuto událost. Aplikace zde ošetřuje vytvoření duplicitní dokumentace nebo pokus uložit dokumentaci s prázdným jménem.



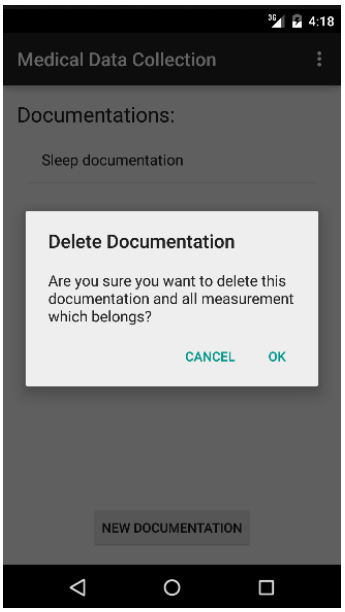
Obrázek 15 Pohled vytvoření nové dokumentace
Zdroj: vlastní

Při úspěšném uložení nové dokumentace se zobrazí v seznamu dokumentací v úvodním pohledu (viz Obrázek 16).



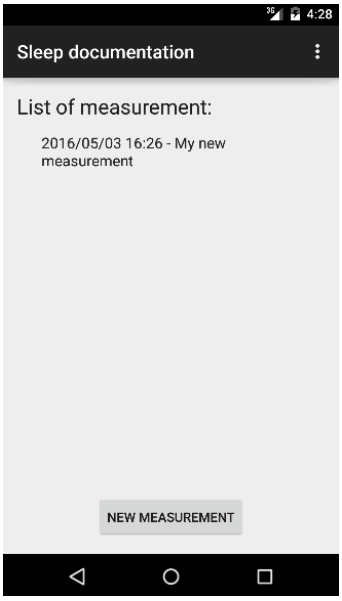
Obrázek 16 Nově vytvořená dokumentace
Zdroj: vlastní

Dále můžeme tuto dokumentaci otevřít jednoduchým kliknutím nebo smazat dlouhým kliknutím na její jméno. Před smazáním dokumentace aplikace vyzve uživatele k potvrzení akce (viz Obrázek 17).



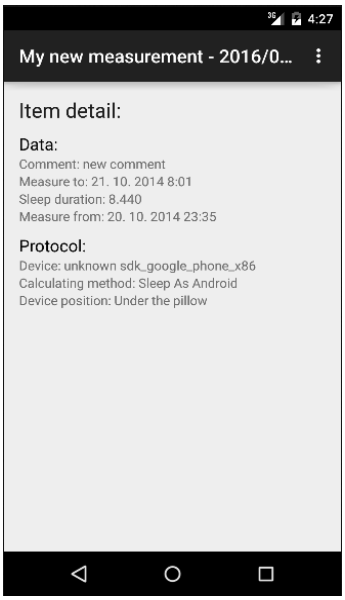
Obrázek 17 Výzva k potvrzení akce smazání dokumentace
Zdroj: vlastní

Při otevření dokumentace se uživateli zobrazí pohled seznamu záznamů z dané dokumentace (viz Obrázek 18).



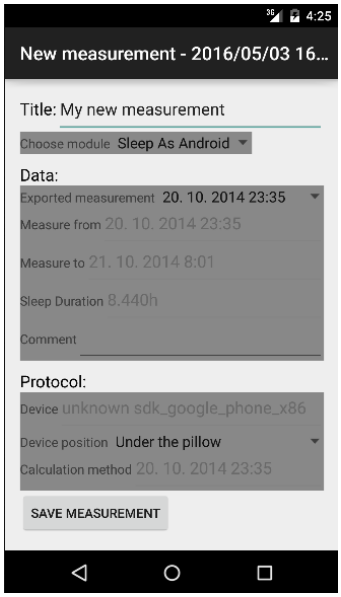
Obrázek 18 Pohled seznamu záznamů
Zdroj: vlastní

Záznamy lze smazat stejným způsobem jako dokumentace, nebo je lze otevřít pro zobrazení jejich detailních informací (viz Obrázek 19).



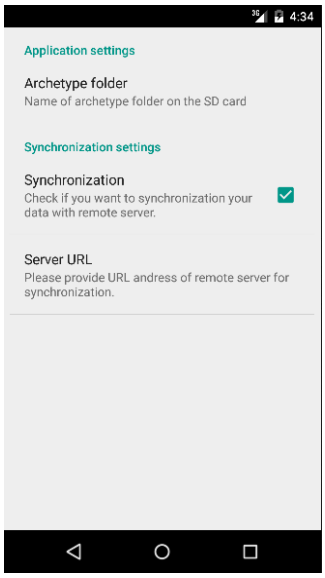
Obrázek 19 Detailní pohled záznamu
Zdroj: vlastní

Pro vytvoření nového záznamu musí uživatel kliknout na tlačítko v dolní části pohledu seznamu záznamů. Po kliknutí se zobrazí pohled pro vytvoření nového záznamu (viz Obrázek 20). V tomto pohledu je možné vybrat modul, který vytváří šablonu pro nový záznam. V rámci této práce byl vytvořen modul pro aplikaci Sleep As Android. Ve Sleep As Android modulu lze vybrat měření, které z této aplikace chce uživatel zaznamenat do dokumentace. Podle této volby se vyplní následující položky. Některé, jako např. komentář lze měnit. Po vyplnění všech položek je možné záznam uložit kliknutím na tlačítko ve spodní části pohledu. Aplikace opět omezuje vytvoření záznamu s duplicitním názvem a vytvoření záznamu s prázdným názvem.

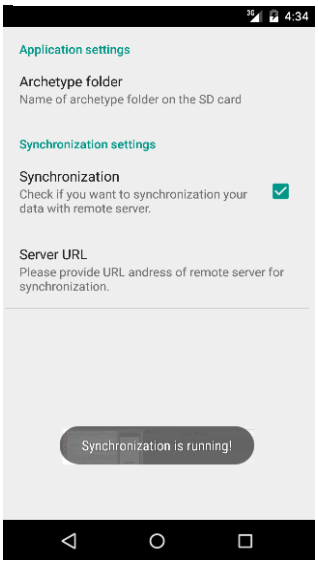


Obrázek 20 Pohled nového záznamu
Zdroj: vlastní

Aplikace může ve všech pohledech zobrazit kontextové menu, ve kterém se mimo jiné nachází položka pro nastavení aplikace (viz Obrázek 21). Kliknutím na tuto položku se uživateli zobrazí pohled nastavení aplikace. V tomto pohledu může uživatel nastavit název složky na SD kartě, ve které se nachází archetypy. Dále je zde kategorie pro synchronizaci aplikace se vzdáleným serverem. V této kategorii se nachází zaškrtávací volba pro zapnutí a vypnutí synchronizace. Při zapnutí synchronizace se aplikace okamžitě snaží navázat spojení se serverem. O stavu synchronizace aplikace uživatele informuje (viz Obrázek 22). Další položka je adresa vzdáleného serveru. Při vyplnění špatné adresy aplikace uživatele upozorní a odmítne synchronizovat.



Obrázek 21 Nastavení aplikace
Zdroj: vlastní



Obrázek 22 Synchronizace úspěšně probíhá
Zdroj: vlastní