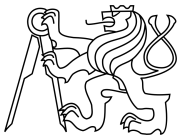
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace PWiL - Systém pro lokalizaci pacientů
Student:	Vít Medřický
Vedoucí:	Ing. Filip Křikava, Ph.D.
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

PWiL je systém umožňující zaměstnancům pečovatelského domu sledovat polohu a stav pacientů pomocí speciálních náramků. Zároveň umožňuje pacientům přivolat si v případě potřeby pomoc. Cílem této práce je webová aplikace, která umožňuje správu systému, zpracovává data z lokalizačního serveru dodaného třetí stranou a poskytuje rozhraní pro mobilní aplikaci.

- Pomocí metod softwarového inženýrství analyzujte požadavky na systém a vyberte vhodné technologie pro jeho realizaci.
- Nastudujte funkcionalitu SPOT API, API náramku a dalších potřebných technologií.
- Na základě předchozího bodu navrhněte API poskytované mobilním zařízením.
- V návaznosti na předchozí body vypracujte návrh systému.
- Navrhněte UI, zamyslete se nad způsoby použití a jak je bude UI pokrývat.
- Implementujte prototyp aplikace skládající se z grafického rozhraní pro uživatele a aplikačního rozhraní pro mobilní zařízení.
- Zhotovený prototyp podrobte vhodnému testování.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 20. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Webová aplikace PWiL - Systém pro lokalizaci pacientů

Vít Medřický

Vedoucí práce: Ing. Filip Křikava, Ph.D.

17. května 2016

Poděkování

Chtěl bych poděkovat mému vedoucímu práce Ing. Filipu Křikavovi, Ph.D. za konstruktivní rady při vývoji aplikace i psaní textu práce. Rád bych také poděkoval všem svým nejbližším za všestrannou podporu, díky které bylo možné tuto práci dokončit.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze, 25. března 2015 dne 17. května 2016 .....

Odkaz na tuto práci

Medřický, Vít. *Webová aplikace PWiL - Systém pro lokalizaci pacientů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá tvorbou webového systému, který umožňuje lokalizaci pacientů v pečovatelském domě. Cílem je usnadnit těmto pacientům přivolat si pomoc v případě nouze. V práci jsou analyzovány požadavky na systém a případy užití, na jejichž základě je navrženo uživatelské rozhraní a schéma jednotlivých stránek aplikace. Je implementován prototyp aplikace v jazyce PHP s využitím frameworku Laravel.

Klíčová slova lokalizační systém Pwil, webová aplikace, SPoT API, návrh uživatelského rozhraní, PHP, Laravel

Abstract

This thesis is focused on the process of creating a web system that makes possible to locate patients in nursing homes. The purpose is to simplify patient's ability to call for help in need. The thesis begins with analysis of system's requirements and use cases, based on which the user interface is designed, and the wireframes are constructed. System is implemented using the Laravel framework.

Keywords location system Pwil, web application, SPoT API, GUI design, PHP, Laravel

Obsah

Seznam obrázků

Úvod

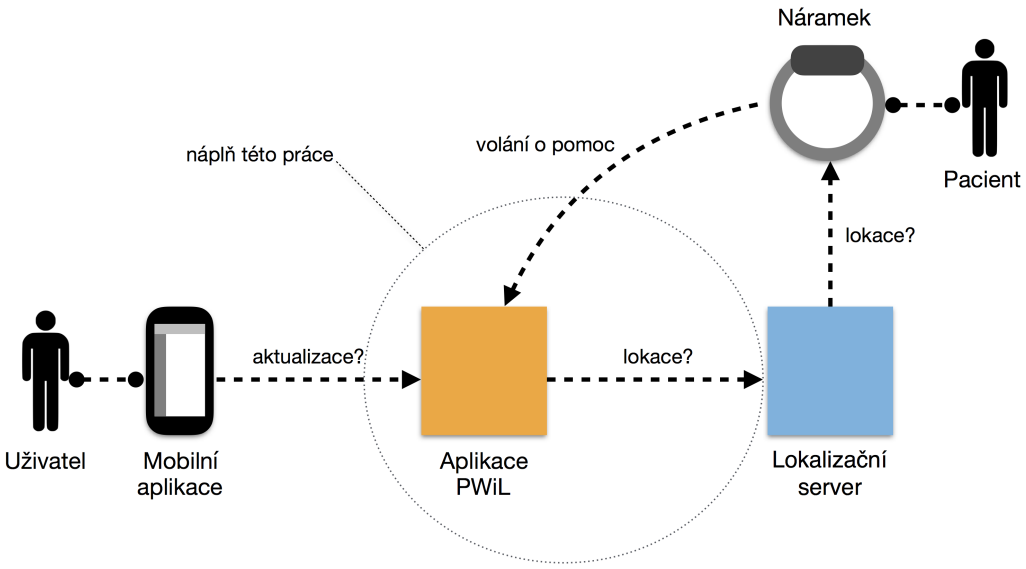
V posledních letech zaznamenaly bezdrátové lokalizační technologie značný pokrok. Většina z nás si už téměř nedokáže představit svět bez GPS či chytrých telefonů s online mapami[1]. Nosíme s sebou telefony s mobilním připojením, díky kterým se můžeme podívat na mapy kdekoliv jsme, či se přímo nechat navigovat k cíli.[2]. Moderní technologie pokročily tolik, že je možné ve venkovních prostorách lokalizovat zařízení s přesností několika metrů[3].

Ústřední roli pro venkovní lokalizaci hraje technologie GPS.[3] Většinu času ovšem lidé tráví uvnitř, kde je tato technologie lokalizace nevhodná. Síla signálu není dostatečná na průnik materiálem našich budov[4]. Možná právě z toho důvodu zaznamenaly v poslední době značný rozvoj i technologie pro lokalizaci uvnitř budov, které v současnosti dokáží určit polohu s přesností na jeden metr[5]. To přináší širokou škálu možností využití v moha odvětvích lidské činnosti. Nabízí se stavebnictví, domácnosti s dětmi, sklady, továrny nebo třeba zdravotnictví.

V nemocnicích či pečovatelských domech, kde je čas velmi důležtý, může lokalizace pomoci zachraňovat životy. Právě domy s pečovatelskou službou, se zabývá tato práce. Je zcela žádoucí vytvořit systém, který by personálu pečovatelského domu usnadňoval lokalizaci pacientů a zároveň zjednodušil pacientům volání o pomoc v případě nouze.

Jedním z pilotních projektů nasazení technologií pro vnitřní navigaci je systém PWiL, který je předmětem této práce. Systém bude využíván zaměstnanci pečovatelských domů k určení přesné polohy pacientů při volání o pomoc a zároveň jako přehled nad jejich pohybem.

Tento systém bude tvořen ze čtyř základních částí:



Obrázek 0.1: Zjednodušené schéma systému

1. **Náramků** - náramky budou pacienti nosit podobně jako například hodinky. Kromě umožnění komunikace s lokalizačním serverem je funkcí náramku možnost přivolat si pomoc stisknutím tlačítka v případě nouze. Náramky budou dodány externí firmou.

2. **Lokalizačního serveru** - tento server bude poskytovat komunikaci mezi náramky a webovou aplikací PWiL. Na základě dotazů webové aplikace bude poskytovat odpověď dle stanoveného protokolu, obsahující informace o lokaci pacienta. Tento server bude dodán jinou externí firmou, která se specializuje na lokaci předmětů v oblasti pokryté bezdrátovou sítí.

3. **Aplikace PWiL** - jejím úkolem je zprostředkovávat komunikaci mezi lokalizačním serverm a mobilní aplikací a zároveň poskytuje možnost administrace systému, tedy správu uživatelů, náramků a záznamů o pacientech a přiřazování náramků k záznamu pacienta, kterému byl náramek přidělen. Tato část je náplní této bakalářské práce.

4. **Mobilní aplikace** - tato mobilní aplikace bude od webové aplikace získávat informace o poloze a stavu ohrožení pacienta a zobrazovat tyto údaje uživateli, tedy pečovateli. Zhotovení této aplikace nespadá do náplně této bakalářské práce.

Cíl práce

Cílem práce je zhotovit prototyp webové aplikace PWiL, který bude schopný spolupracovat s ostatními komponentami systému. Systém bude zaměstnancům pečovatelského domu usnadňovat lokaci pacientů. Pacientům bude umožňovat, aby si přivovali pomoc v jakékoliv situaci, bez nutnosti hledání nejbližšího poplašného vypínače. Webová aplikace PWiL zde bude sehrávat pozici prostředníka, který je zcela nezbytný pro chod systému a zároveň bude sloužit jako ústřední komponenta pro administraci systému.

Motivace

Základní motivací pro výběr tohoto tématu je zajisté možnost pomoci lidem. To jak v případě zaměstnanců pečovatelských domů, tak i pacientů. Zaměstnanci díky této technologii získají lepší přehled nad svými pacienty a pacientům bude díky tomuto systému poskytnuta kvalitní péče. Neméně zajímavou motivací je poznání nových, moderních technologií v oblasti webových služeb a zároveň zcela unikátních technologií v oblasti lokalizace.

Kapitola 1

Analýza

V této kapitole shrneme analýzu projektu. Začneme s funkčními a nefunkčními požadavky systému PWiL, dále zmíníme jednotlivé role a případy užití systému.

1.1 Funkční požadavky

Funkční požadavky popisují chování a funkčnost, tedy co by měl systém umět. Tyto požadavky na systém PWiL jsou následující:

1. **Přihlášení:** Systém bude z důvodu bezpečnosti obsahovat autentikaci pro uživatele. Každý uživatel bude mít vlastní přihlašovací údaje, pomocí kterých se bude moci přihlásit.

2. **Správa uživatelů:** V souvislosti s předchozím bodem bude nutné aby systém obsahoval uživatelské účty. Bude možné tyto účty přidávat, odebírat, případně upravovat.

3. **Správa pacientů:** Systém bude umožňovat přidávání a odebírání pacientů a změnu jejich údajů.

4. **Správa náramků:** Systém dále umožní přidávání a odebírání lokalizačních náramků. Tyto náramky bude možné spojit s konkrétním záznamem pacienta. Bude možné toto spojení upravovat, tedy přiřadit náramek jinému pacientovi. Pacient bude mít současně přiřazený vždy maximálně jeden náramek.

5. **Notifikace:** Systém bude umět mobilní aplikaci předat informaci o pacientově volání o pomoc. To bude zprostředkované zmáčnutím tlačítka na náramku. Pečovatel poté bude upozorněn přes mobilní aplikaci, která bude se systémem komunikovat.

6. **Načítání dat ze SPoT API:** Systém bude umět načítat data o pozici pacientů z lokalizačního serveru přes SPoT API.

7. **Sledování lokace pacientů:** V systému bude umožněno sledovat, kde se pacienti aktuálně nachází.

8. **Příjem informací od náramku:** Systém bude od náramku schopen obdržet informaci o volání o pomoc.

9. **Poskytování dat mobilní aplikaci:** Systém bude umět data z předchozích dvou bodů poskytovat mobilní aplikaci.

1.2 Nefunkční požadavky

Nefunkční požadavky popisují „jak“ by měl systém fungovat. Dá se na ně nahlížet jako na atributy kvality systému.[6] Pro tento systém to jsou:

1. **Spolehlivost a dostupnost:** V systému, který může člověku zachránit život je velmi důležité, aby se na něj dalo spolehnout. Je potřeba předejít výpadkům a vyhnout se nespolehlivým technologiím a zajistit tak maximální dostupnost. Nespolehivý systém v této oblasti znamená větší hrozbu, než žádný systém.

2. **Bezpečnost dat:** Systém bude uchovávat osobní údaje o uživatelích, osobní a lokační údaje o pacientech. V budoucnu je možné, aby systém uchovával i další osobní údaje. Tato data by měla být dostatečně zabezpečena, aby se předešlo jejich úniku do nepověřených rukou.

3. **Rozvoj a škálovatelnost:** V budoucnu se systém bude rozšiřovat a další funkcionality ja pravděpodobné, že na něm budou prováděny změny. Je žádoucí, aby tyto úpravy byly jednoduše proveditelné a aby se systém dal jednoduše rozšiřovat a vylepšovat.

1.3 Role v systému

1.3.1 Uživatelské role

Sytém bude zabezpečen přihlašovací bránou, přes kterou se dostane uživatel pouze po zadání svých uživatelských údajů. Měl by podporovat uživatelské role. Ty určují, jaké akce smí uživatel dělat. Ze začátku máme jen dvě role, do budoucna je ovšem možné, že přibudou další funkce. S tím je dobré počítat při návrhu systému.

- **Uživatel:** Uživatelem je zaměstnanec pečovatelského domu (pečovatel). K systému má přístup přes webovou aplikaci, nebo přes aplikaci na mobilním zařízení.

- **Administrátor:** Administrátorem je správce systému. Má veškeré pravomoce uživatele, navíc však může spravovat uživatelské účty.

1.3.2 Další role

Aktéry nemusí být jen osoby. V našem případě máme externí systémy, které užívají náš systém. Ty je třeba zmínit:

- **Mobilní aplikace** - která se bude systému dotazovat na seznam pacientů a jejich změny v lokaci.
- **Náramek** - bude systému zasílat volání o pomoc.
- **Subsystém komunikující s lokalizačním serverem** - bude se dotazovat *lokalizačního serveru* na lokace pacientů a ukládat jeho odpovědi do databáze.

1.4 Případy užití

Případy užití popisují použití systému jednotlivými aktéry (osobami, či systémy). Zároveň jsou popsány reakce systému. V následující sekci si tyto *případy užití* přiblížíme.

1.4.1 Případy užití pro uživatele

- **Přihlášení do systému:**
  - **Scénář I:**
    1. Uživatel vstoupí na stránku s přihlášením.
    2. Uživatel vyplní přihlašovací údaje a potvrdí tlačítkem přihlásit.
    3. Uživateli se podaří přihlásit, zobrazí se mu úvodní stránka aplikace.
  - **Scénář II:**
    1. Uživatel vstoupí stránku s přihlášením.
    2. Uživatel vyplní přihlašovací údaje a potvrdí tlačítkem přihlásit.
    3. Uživateli se nepodaří přihlásit, zobrazí se mu upozornění, že přihlašovací údaje jsou nesprávné.
    4. Uživatel klikne na tlačítko "zapomněl jsem heslo".
    5. Uživatel zadá svůj email a nové heslo.
    6. Uživateli přijde email s novým heslem.
    7. Uživatel zopakuje přihlášení.
    8. Uživateli se podaří přihlásit, zobrazí se mu úvodní stránka aplikace.

- **Přidání pacienta:**
  - **Scénář I**
    1. Uživatel vstoupí na stránku s přidáním pacienta.
    2. Uživatel vyplní údaje pacienta.
    3. Uživatel potvrdí vyplněné údaje.
    4. Uživateli se zobrazí stránka s přehledem pacientů, včetně nového pacienta.
  - **Scénář II**
    1. Uživatel vstoupí na stánku s přidáním pacienta.
    2. Uživatel nevyplní všechny potřebné informace.
    3. Uživatel bude upozorněn na špatně vyplněné údaje a uložení se neprovede.
- **Upravení údajů pacienta:**
  - **Scénář I**
    1. Uživatel vstoupí na stránku s úpravou pacienta.
    2. Uživatel změní údaje pacienta.
    3. Uživatel potvrdí změny.
    4. Uživateli se zobrazí stránka s upraveným pacientem.
  - **Scénář II**
    1. Uživatel vstoupí na stránku s úpravou pacienta.
    2. Uživatel smaže některé z údajů.
    3. Uživatel potvrdí změny.
    4. Uživatel bude upozorněn na špatně vyplněné údaje a změny se neprovedou.
- **Odebrání pacienta:**
  - **Scénář I**
    1. Uživatel vstoupí na stránku s úpravou pacienta.
    2. Uživatel zmáčkne tlačítko `Smazat pacienta`.
    3. Uživatel bude dotázán, zda si je smazáním jistý.
    4. Uživatel smazání potvrdí.
    5. Uživateli se zobrazí stránka se seznamem pacientů bez smazaného pacienta.
  - **Scénář II**
    1. Uživatel vstoupí na stránku s úpravou pacienta.
    2. Uživatel zmáčkne tlačítko `Smazat pacienta`.

3. Uživatel bude dotázán, zda si je smazáním jistý.
4. Uživatel zmáčkne tlačítko `Zrušit`.
5. Uživatel zůstane na stránce s úpravou pacienta a smazání neproběhne.

- **Zjištění lokace pacienta:**

1. Uživatel vstoupí na stránku s přehledem pacientů.
2. Uživatel se podívá na aktuální pozici pacienta ve sloupci *lokace*.

- **Zjištění stavu pacienta:**

1. Uživatel vstoupí na stránku s přehledem pacientů.
2. Uživatel se podívá na aktuální stav pacienta ve sloupci *stav*.

- **Zjištění dalších údajů o pacientovi:**

1. Uživatel vstoupí na stránku s přehledem pacientů.
2. Uživatel klikne na jméno požadovaného pacienta.
3. Uživateli se zobrazí detail vybraného pacienta s potřebnými detaily.

Přidání, úprava a odebrání náramků proběhne podle stejného scénáře, pouze s jinými atributy. Při analýze projektu byly i tyto případy užití zhotoveny, zde si můžeme dovolit je přeskočit.

- **Přiřazení náramku k pacientovi:**
  - **Scénář I**
    1. Uživatel vstoupí na stránku s párováním.
    2. Uživatel vybere pacienta.
    3. Uživatel vybere náramek.
    4. Uživatel potvrdí spojení.
    5. Uživateli se zobrazi potvrzení o provedeném spojení.
  - **Scénář II**
    1. Uživatel vstoupí na stránku s párováním.
    2. Uživatel nevyplní obě výběrová pole.
    3. Uživatel potvrdí spojení.
    4. Uživatel bude upozorněn, že nevyplnil obě pole, přiřazení neproběhne.

- **Odhlášení:**

1. Uživatel klikne v pravém horním rohu aplikace na své jméno.
2. Uživateli se zobrazí možnost odhlášení.

3. Uživatel zmáčkne tlačítko odhlásit se.
4. Uživateli bude odhlášen a zobrazí se mu přihlašovací stránka.

Případy užití pro administrátora jsou stejné jako pro uživatele. Přibudou zde scénáře pro přidání, odebrání a úpravu uživatele, které jsou opět téměř identické se scénáři nad pacienty, byly tedy vytvořeny, ale nebudeme je úvadět.

1.4.2 Případy užití pro mobilní aplikaci

- **Dotaz na seznam pacientů:**
  - **Scénář I**
    1. Mobilní aplikace zašle validní[1] požadavek na stanovenou adresu pro výpis pacientů.
    2. Systém odpoví výpisem se seznamem aktuálních pacientů.
  - **Scénář II**
    1. Mobilní aplikace zašle nevalidní požadavek.
    2. Systém odpoví příslušnou chybovou hláškou.
- **Dotaz na změny:**
  - **Scénář I**
    1. Mobilní aplikace zašle validní požadavek na stanovenou adresu pro výpis změn lokací a stavu pacientů. Mobilní aplikace v požadavku uvede od jakého času chce změny.
    2. Systém odpoví výpisem stránky se seznamem posledních změn.
  - **Scénář II**
    1. Mobilní aplikace zašle nevalidní požadavek, nebo nevyplní údaje o času od kterého chce změny.
    2. Systém odpoví příslušnou chybovou hláškou.

1.4.3 Případy užití pro náramek

- **Zaslání hlášení o volání o pomoc:**
  1. Náramek zašle hlášení o volání o pomoc na stanovenou adresu.
  2. Aplikace hlášení zpracuje.

[1] Bude splňovat definici požadavku stanovenou naším rozhraním.

1.4.4 Případy užití pro subsystém komunikující s lokalizačním serverem

- **Zaslání dotazu na lokaci pacientů:**
- **Dotaz na změny:**
  - **Scénář I**
    1. Subsystém zašle požadavek na poslední známé lokace pacientů.
    2. Lokalizační server odpoví ve formátu dle dokumentace.
    3. Subsystém uloží odpověď do databáze.
  - **Scénář II**
    1. Subsystém zašle požadavek na poslední známé lokace pacientů.
    2. Lokalizační server neodpoví ve formátu dle dokumentace, nebo neodpoví vůbec.
    3. Subsystém nahlásí chybu v komunikaci.[2]

1.5 Rozhodnutí použitého frameworku

Pro implementaci tohoto projektu bylo možné vybírat ze široké škály *frameworků*[3]. Aby byl projekt časově realizovatelný, bylo potřeba vybrat technologii, kterou se lze snadno naučit. Zadavatelem projektu byl pro zhotovení požadován PHP framework Laravel, ve kterém zadavatelská firma vyvíjí další projekty. Navíc bylo autorovi nabídnuto doučování tohoto frameworku. Laravel zároveň částečně vychází z frameworku Symfony2, se kterým se autor setkal v předmětech BI-WT1 a BI-WT2[7], a tak jeho použití vyhovělo oběma stranám.

[2] Zde je potřeba se zamyslet nad efektivním způsobem hlášení chyb - například zaslání emailu po několika nepovedených pokusech.

[3] Cílem frameworku je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání.

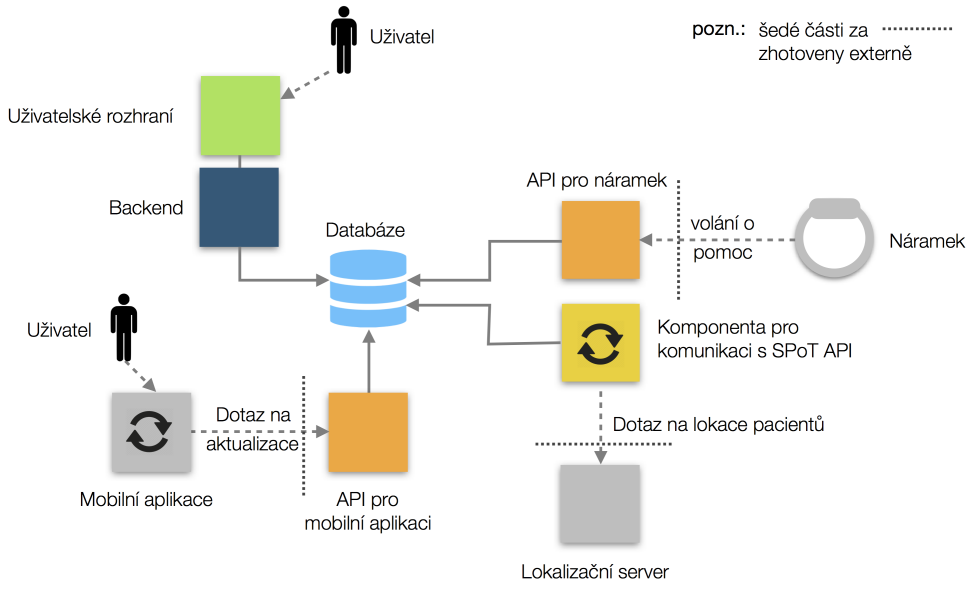
Kapitola 2

Návrh

Tato kapitola se zabývá návrhem jednotilvých částí systému na základě poznatků, které vzešly z analýzy.

2.1 Návrh struktury systému

Pro základní orientaci v systému musela být nejprve navrhnuta jeho struktura, která je detailnějším zobrazením obecného schématu z úvodu (obr.0.1).



Obrázek 2.1: Struktura systému

Návrh dílčích částí tohoto schématu rozebereme v následujících sekcích.

2.2 Návrh datového úložiště

V případě, že aplikace pracuje s větším možstvím dat, například při správě několika entit, je zcela vhodné mít tato data uložená v databázi. Jedná se o klasický problém správy strukturovaných dat, pro které byly databáze určeny[8].

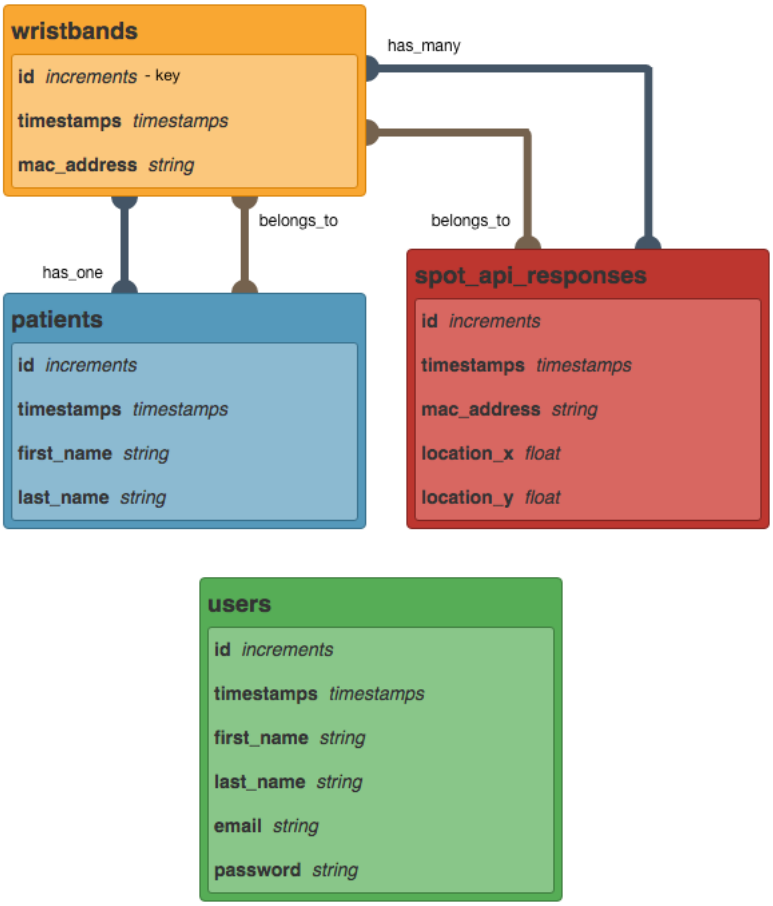
Databáze přináší pro správu dat hned několik výhod, vyjmenujme si ty nejpodstatnější pro náš systém:

- Dotazy - pomocí řetězení dotazů lze z databáze získat potřebné informace. Například všechny uživatele, kteří se jmenují Jiří, nebo náramky, které byly přidány do systému minulou středu.
- Podpora simulačního přístupu více uživatelů - v aplikaci PWiL bude několik uživatelů.
- Bezpečné provádění změn - veškeré změny jsou prováděny pomocí *transakcí*, což umožňuje úpravu dat vrátit při chybném provedení změn.[11]
- Většina frameworků pro tvorbu webových aplikací má nativní podporu databází - tím je i námi použitý Laravel.

2.2.1 Návrh databázového modelu

Na základě analýzy už máme základní představu o složení dat, se kterými bude aplikace pracovat. Pro sestvení databáze je ale nutné vytvořit si její návrh. To se klasicky dělá vytvořením ER diagramu.[8]

Obrázek 2.2 představuje zjednodušený ER diagram[4] doménového modelu aplikace PWiL.



Obrázek 2.2: ER diagram - doménový model

- **Tabulka pacientů** (`patients`) - v této tabulce budou uloženy záznamy o pacientech, tedy jejich jméno a příjmení.

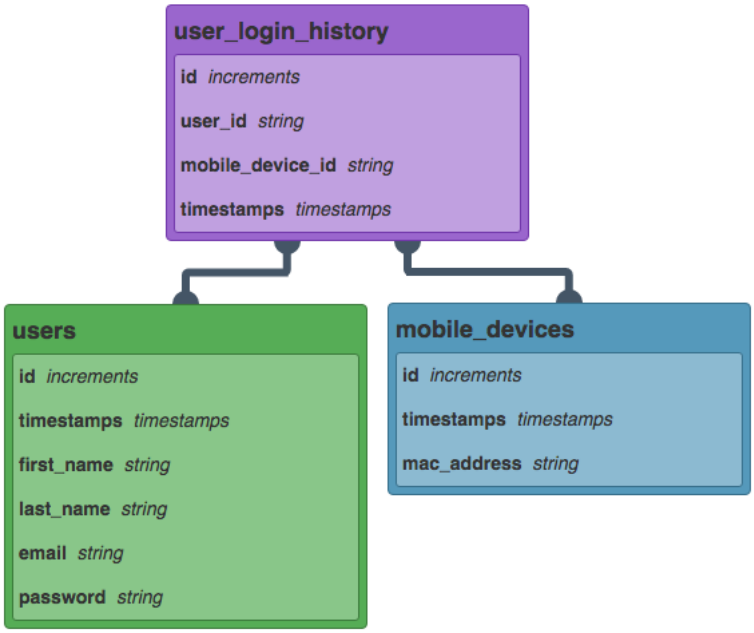
- **Tabulka náramků** (`wristbands`) - tato tabulka obsahuje záznamy o náramcích. Jediným potřebným atributem je jejich MAC adresa.

[4] K vytvoření tohoto diagramu byl použit nástroj Laravel Schema Designer[12], protože z něho můžeme rovnou vygenerovat kód pro modelovou vrstvu.

- **Tabulka odpovědí lokalizačního serveru** (`spot_api_responses`) - v této tabulce budou uloženy odpovědi lokalizačního serveru. Aplikace bude tyto data později používat k vytvoření odpovědi pro mobilní aplikaci a zároveň tabulka poslouží jako historie odpovědí při komunikaci se serverem.

- **Tabulka uživatelů** (`users`) - v poslední tabulce budou uloženi uživatelé. Je třeba evidovat jejich jméno, příjmení, e-mailovou adresu a heslo. Heslo bude ukládáno šifrovaně kvůli bezpečnosti.

  Tato tabulka může být rozšířena o historii přihlášení. Bude tak možné sledovat, který uživatel byl kdy přihlášen na daném mobilním zařízení. Schéma tohoto vztahu je navrženo následnovně:



Obrázek 2.3: Databázový model - historie přihlášení

Tato část databáze nebude v prototypu implementována, protože pro výslednou funkčnost aplikace nehraje podstatnou roli.

2.3 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní je velmi důležitou součástí aplikace z hlediska její použitelnosti. Nejde o pouhý proces sestavení součástí systému. Uživatelské rozhraní se zaměřuje na jednoduchost, přehlednost, jednoznačnost a srozumitelnost. Jeho cílem je předvídat, co by uživatel mohl potřebovat v systému udělat. Poskytovat nástroje a prvky, aby těchto potřeb mohl uživatel co nejjednodušeji dosáhnout[9]. Aby tyto kroky činil intuitivně. „Nejlepší uživatelské rozhraní je takové, kterého si člověk nevšimne“ - tedy uživatelské rozhraní, díky kterému se uživatel může soustředit pouze na své cíle, aniž by byl rozptylován jeho špatným návrhem[10].

2.3.1 Wireframe

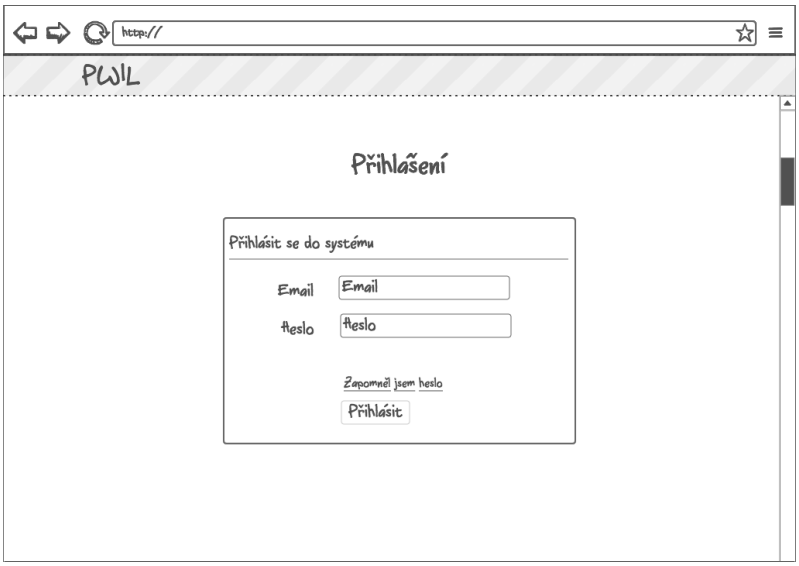
Vytvoření *wireframu* je vhodným nástrojem pro představu podoby jednotlivých stránek aplikace. Jedná se o načrtnutí aplikace, které slouží jako vizualizace rozložení prvků na stránkách a jejich funkcí.

Pro návrh správného modelu je potřeba vyjít z případů užití uvedených v předchozí kapitole. Jednotlivé stránky musí pokrývat všechny tyto případy. Z návrhu bude navíc zřejmé, jakým způsobem bude aplikace používána. Dobrý wireframe je velmi užitečným stavebním kamenem pro implementaci.

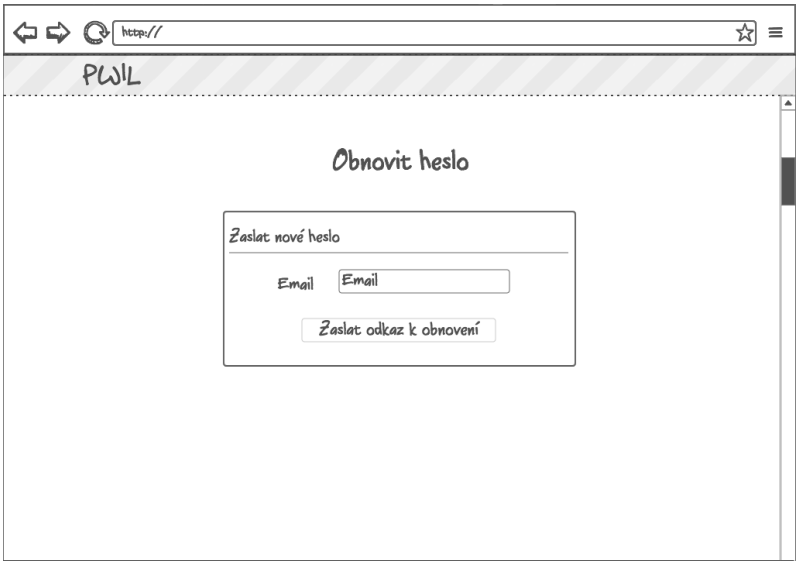
Následující část práce obsahuje wireframy webové aplikace PWiL.

2.3.1.1 Přihlášení

Aplikace PWiL bude přístupná pouze pro přihlášené uživatele. Proto je potřebné mít v systému stránku, na které bude umožněno přihlášení. Tato stránka by měla být první stránkou systému, kterou nepřihlášený uživatel uvidí. Teprve po přihlášení bude vpuštěn do systému.



Obrázek 2.4: Přihlášení

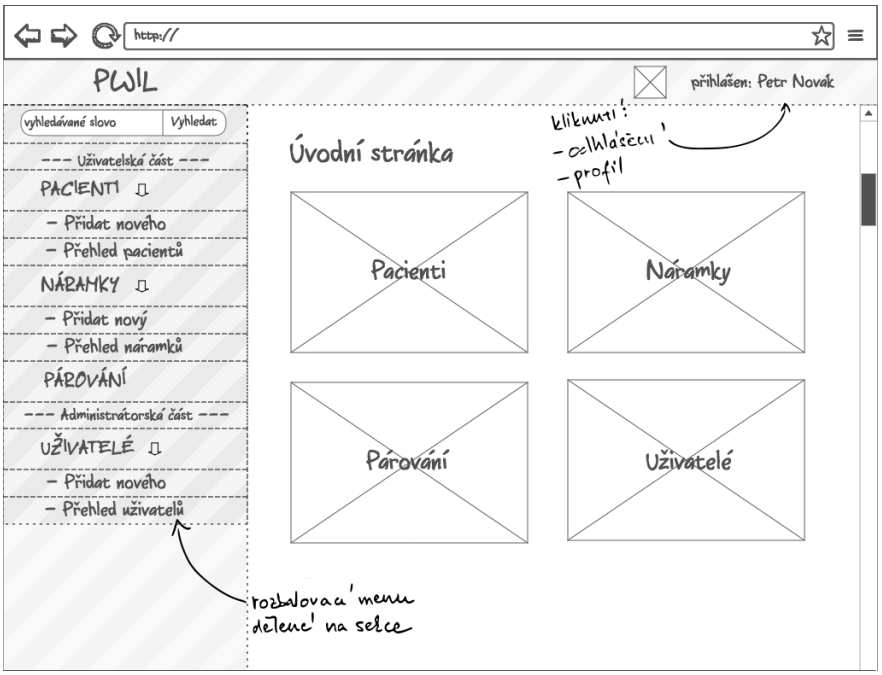


Obrázek 2.5: Zapomenuté heslo

2.3.1.2 Úvodní stránka

Tato stránka je domovskou stránkou aplikace. Uživatel by zde měl najít veškeré funkce, které může potřebovat a měl by se umět rychle zorientovat. Přidanou hodnotou této stránky může být například výpis aktuálních počtů registrovaných pacientů, náramků nebo uživatelů, či další vhodné aktuality.

Součástí tohoto wireframu je i schéma boční nabídky, která se bude opakovat v celém systému (kromě přihlášení). Z důvodu přehlednosti ale už nebude na dalších wireframech znázorněn její detail.

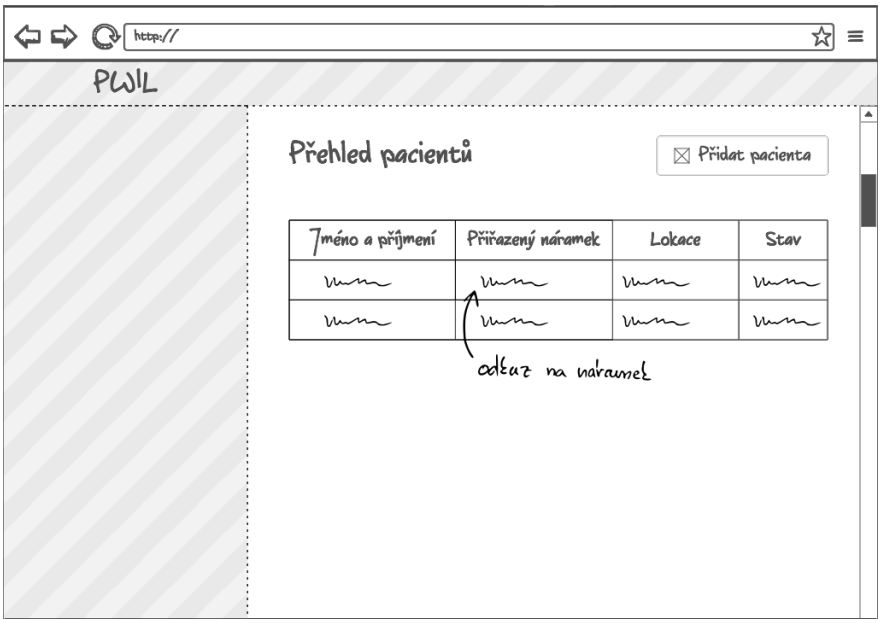


Obrázek 2.6: Úvodní stránka

2.3.1.3 Zjednodušení

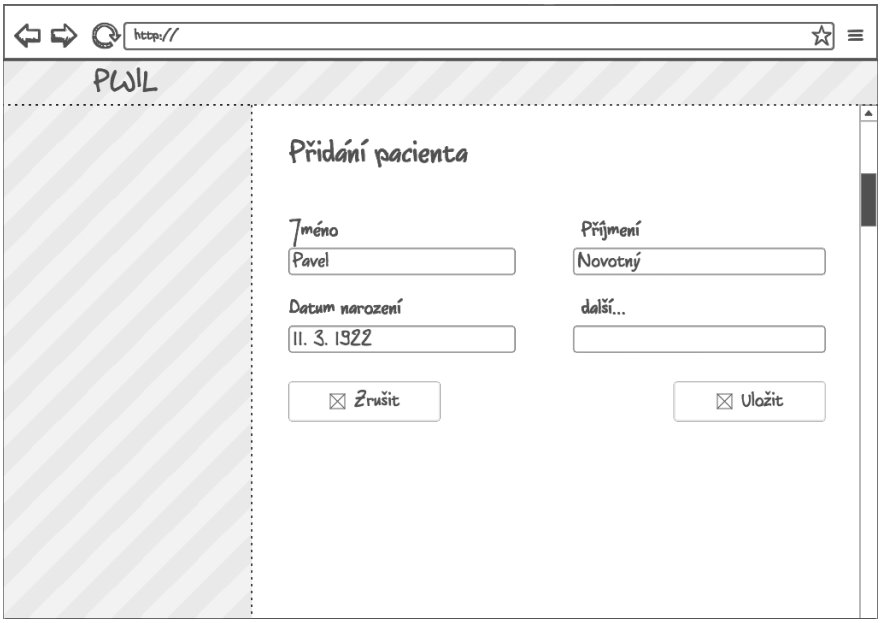
V aplikaci se budou opakovat tři velmi podobné entity: pacienti, náramky a uživatelé. Vzhledem k vysoké podobě operací, které nad nimi budou prováděny, si vybereme jen jednu entitu - pacienty - jako vzorovou a na ní si ukážeme jednotlivé wireframy.

2.3.1.4 Přehled pacientů



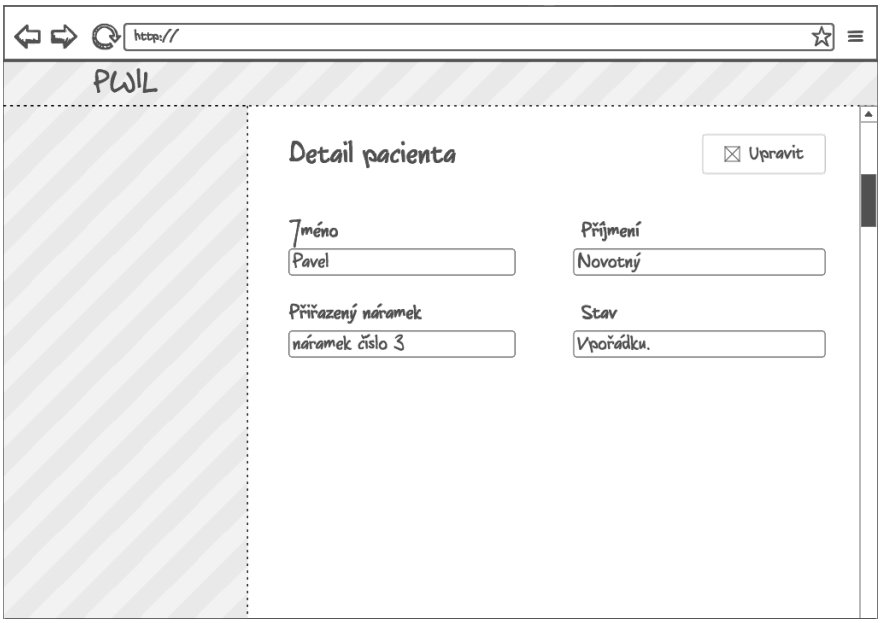
Obrázek 2.7: Přehled pacientů

2.3.1.5 Přidání pacienta



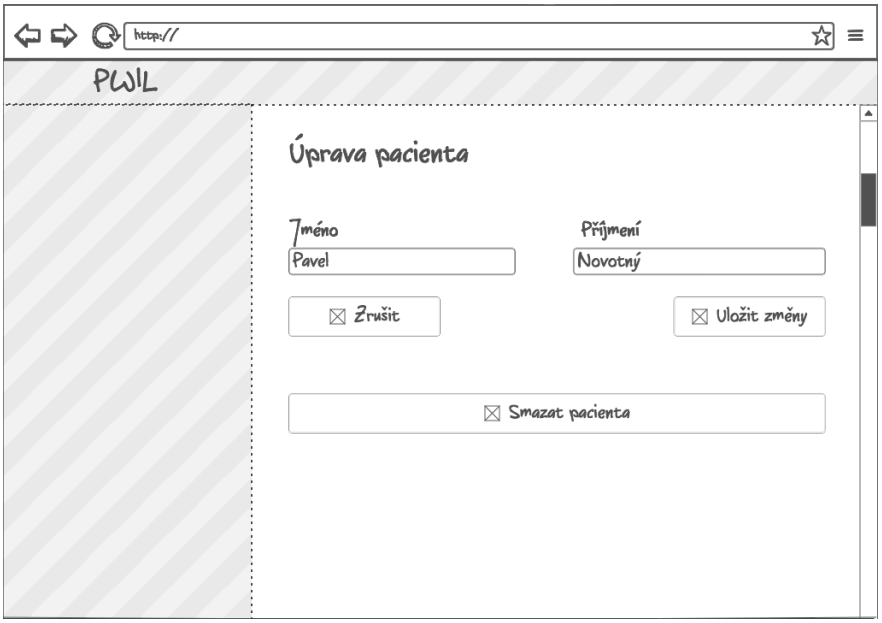
Obrázek 2.8: Přidání pacienta

2.3.1.6 Detail pacienta



Obrázek 2.9: Detail pacienta

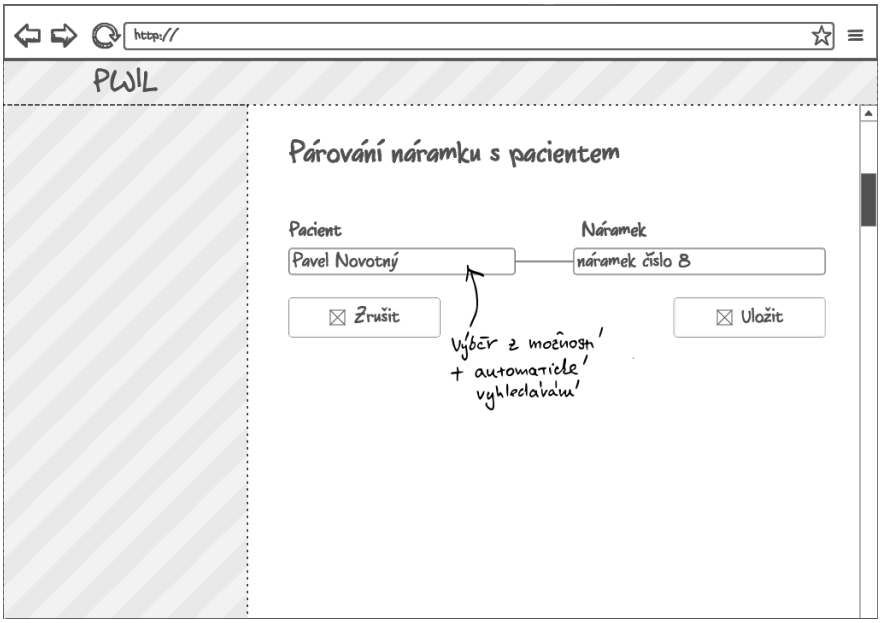
2.3.1.7 Úprava údajů o pacientovi



Obrázek 2.10: Úprava pacienta

2.3.1.8 Párování s náramkem

Pro spárování náramku s pacientem bude samostatná stránka. Uživatel bude moci z nabídky vybrat, či vyhledat požadovaného pacienta a vybrat k němu volný náramek. Uživatel by měl dostat nejdříve na výběr z volných náramků, poté teprve z obsazených, které bude možné přeřadit. V takový okamžik by měl být uživatel upozorněn, že odebírá náramek jinému pacientovi.



Obrázek 2.11: Párování

2.3.2 Použitý nástroj

Tyto wireframy byly navrženy pomocí bezplatného online nástroje NinjaMock[13]. Tento nástroj se osvědčil jako užitečný pro dosažení našeho cíle, nicméně wireframy byly ještě později graficky upraveny. Tento online nástroj totiž nenabízel dostatečné možnosti pro nastavení rozměrů okna.

K vytvoření wireframů byly vyzkoušeny i další online i desktop aplikace. Základními problémemy těchto dalších nástrojů bylo buď příliš široké zaměření - LucidChart[14], placená licence[5] - Balsamiq Mockups 3[15], nebo příliš vysoký detail a profesionalita - Justinmind[16][6]

Určitě nebyly vyzkoušeny všechny možné nástroje, NinjaMock byl pouze prvním, který splnil svůj účel.

[5]Tato licence byla na měsíc zdarma, nicméně při požadavku na úpravu wireframů po 5 týdnech od prvního vyzkoušení aplikace byla už nebylo možné program použít.

[6]Tento nástoj je určený spíše pro detailní mockupy stránek včetně jejich grafické podoby.

2.4 Návrh API

Pro správnou komunikaci s mobilní aplikací a získávání informací od náramků je třeba navrhnout způsob, jakým budou tyto komponenty mezi sebou komunikovat. Je třeba stanovit si rozhraní, které budou aplikace používat pro předávání potřebných informací. Vhodná technologie pro tuto komunikaci je přes REST API, které umožňuje komunikaci pomocí jednoduchých HTTP dotazů[17].

Tato komunikace fuguje tak, že klient (v našem případě mobilní aplikace nebo náramek) zašle dotaz (HTTP požadavek) na předem smluvenou adresu. Na tento požadavek mu poskytovatel (aplikace PWiL) odpoví dle stanoveného protokolu. Jaký tvar požadavku klient zašle, na jakou konkrétní adresu, a současně v jakém formátu mu bude odpovězeno; je otázka předem domluveného, jasně definovaného rozhraní.

Tato aplikace řeší komunikaci přes API s dvěma klienty:

- s náramky
- s mobilní aplikací

V následujících části si tyto dva klienty rozebereme podrobněji.

2.4.1 Komunikace s náramky

Náramky zasílají na specifickou adresu naší aplikaci informaci o volání o pomoc, tedy o zmáčknutí tlačítka. Je třeba jasně stanovit, na kterou adresu bude náramk posílat HTTP požadavek. V tomto případě se jedná o takzvaný POST požadavek, tedy odeslání dat příjemci ke zpracování.

Na základě analýzy a dostupných informacích o náramku, bylo navrženo rozhraní v tomto tvaru:

```
POST    /api/v1/wristbands-communication
Vstupní parametry
mac_address
     Required String - MAC adresa náramku

Vzorová odpověď
200 OK
```

Obrázek 2.12: API pro náramek

Aplikace PWiL tedy identifikuje náremek podle jeho MAC adresy, která byla náramku přidělena při přidání do systému. Webová aplikace přijme HTTP požadavek a zpracuje informaci jako volání o pomoc.

2.4.2 Komunikace s mobilní aplikací

Mobilní aplikace potřebuje od aplikace PWiL získávat aktuální informace o polohách pacientů a jejich hlášení o pomoc. Tyto informace budou předávány na jiné adrese API.

Na základě osobní schůzky se stranou zhotovující mobilní aplikaci bylo domluveno následující rozhraní.

2.4.2.1 Výpis pacientů

Toto schéma popisuje dotaz na seznam pacientů. Mobilní aplikace potřebuje tuto informaci, aby měla přehled o seznamu pacientů.

GET /api/v1/patients

Vzorová odpověď

```
200 OK

[
  {
    “id”: 1,
    “first_name”: “Ladislav”,
    “last_name”: "Hruška",
  },
  {
    “id”: 3,
    “first_name”: “Božena”,
    “last_name”: "Němcová",
  },
…
]
```

Obrázek 2.13: API pro mob. aplikaci - výpis pacientů

Pomocí tohoto protokolu obdrží mobilní aplikace seznam pacientů a jejich atributů, které poté může dále využívat.

2.4.2.2 Výpis změn

Hlavním bodem komunikace mezi těmito dvěma komponentami je výpis změn. Mobilní aplikace zažádá o výpis změny v lokacích pacientů a volání o pomoc. Jako odpověď se jí dostane výpis pouze těch pacientů, jejichž lokace se změnila, a seznam těch, kteří volají o pomoc.

Zároveň ve výpisu bude identifikátor změn v evidenci pacientů. Tedy jestli nebyl přidán, upraven, nebo odebrán některý z pacientů. V případě, že tyto změny nastanou, mobilní aplikace zažádá o výpis pacientů podle předchozího protokolu.

GET /api/v1/digest

Vstupní parametry

from
  Required **Timestamp** - čas poslední známé změny mobilní aplikací

till
  Required **Timestamp** - aktuální čas mobilní aplikace

Vzorová odpověď

200 OK

```
{
  “location_changes”: {
    “1”: {
      “location_x”: “100.85”,
      “location_y”: "600.39"
    },
    “3”: {
      “location_x”: “120.85”,
      “location_y”: "230.39"
    }
  },
  “patient_changes”: {
    false
  },
  “calling_for_help: {
    1, 3
  }
}
```

Obrázek 2.14: API pro mob. aplikaci - výpis změn

Při navrhování bylo uvažováno i nad jinak strukturuovaným výpisem změn, který by obsahoval seznam všech pacientů, u kterých nastala *libovolná* změna.

Nevýhodou tohoto řešení je, že jeho proces je ucelený a složitěji se do něj zasahuje. Vzhledem k plánovanému rozšíření tohoto výpisu do budoucna byla tedy upřednostněna varianta rozdělení na jednotlivé celky. V takovém řešení se dá k jednotlivým zpracovávaným údajům přistupovat samostatně a jsou na sobě nezávislé. Jejich budoucí úprava, či rozšíření bude tedy jednodušší a přehlednější.

2.5 Komunikace s lokalizačním serverem

Obdobnou komunikací, kterou tato aplikace řeší je komunikace s lokalizačním serverem. V této komunikaci je ale klientem naše webová aplikace a tudíž návrh rozhraní je poskytnut z druhé strany - v dokumentaci SPoT API od firmy Ruckus Wireless.

Komunikace bude fungovat tak, že aplikace PWiL se bude serveru dotazovat na poslední známé polohy pacientů. Jako odpověď dostane seznam pacientů s posledním známými lokacemi.

GET /venues/:venue_id/locations/last_known.json

Input Parameters

seconds_ago

Optional **Integer** - Number of seconds to look back, limited to **300**. Default: **60**.

macs[]

Optional **String** - MAC addresses to filter by, in this format: **macs[]=00000000000&macs[]=FFFFFFFFFFFF**. Default: all MAC addresses.

Example Response with Hashing and MAC Filter

200 OK

```
[
  {
    "mac": "2CE6CC6D67F8",
    "timestamp": "2013-07-18T09:59:00+08:00",
    "floor_number": 1,
    "x": 500.8518743694843,
    "y": 600.3700100882511,
    "located_inside": true,
    "zones": [
      {
        "zone_map_name": "section",
        "zone_name": "north_wing"
      },
      {
        "zone_map_name": "department",
        "zone_name": "marketing"
      }
    ]
  }
]
```

Obrázek 2.15: SPoT API - výpis posledních známých pozic

Vzhledem k požadavkům aplikace na spolehlivost se tyto dotazy budou opakovat přibližně 1x za sekundu. Je žádoucí mít průběžný přehled o pohybu pacienta, nikoliv s časovými skoky.

Rozhraní komunikace, definované podle dokumentace SPoT API, nabízí další možnost lokalizace. Tou je určení polohy v domě pomocí názvu daného sektoru, či místnosti. Toto nastavení počítá s instalací lokalizačního serveru do konkrétní budovy. S těmito nastaveními bude prototyp počítat do budnoucna, zatím bude ale pracovat pouze se souřadnicemi.

Kapitola 3

Implementace

Tato kapitola popisuje implementaci aplikace PWiL. Jelikož se jedná o větší projekt, v rámci této práce byl implementován pouze její prototyp se základní funkčností.

Cílem této kapitoly je popsat postup při implementaci tohoto prototypu, který bude tvořit jádro výsledného systému. Prototyp této aplikace tedy počítá s rozšiřováním a následným nasazením na trh.

Jednotlivé kroky implmenentace vycházejí z předchozích kapitol. Byly použity poznatky nabyté při analýze a bude se vycházet z návrhů jedotlivých částí aplikace.

3.1 Použité technologie

K implementaci prototypu se rozhodlo pro použití PHP frameworku Laravel, který je postaven na návrhovém vzoru MVC[18]. V následujících sekcích budeme komentovat jeho jednotlivé části. Přiblížíme jeho funkcionalitu a roli, kterou hraje při vytváření takovýchto projektů.

Podle návrhu datového úložiště v předchozí kapitole byla impelementována databáze systému. Z použitého nástroje Laravel Schema Designer byly vygenerovány následující části kódu:

- **Databázové migrace** - tyto migrace slouží k vytvoření tabulek v databázi. Pokud je tedy nutná nějaká změna v databázi, přepíše se pouze předpis migrace a pomocí nástroje Artisan[7] se provede migrace databáze.

- **Modely** - pomocí modelů je umožněn snažší přístup pro práci s databází. Každou tabulku databáze zastupuje vlastní model. Tyto modely

[7] nástroj Laravelu pro interní funkce - usnadňuje práci pomocí jednoduchých příkazů v konzoli

umožňují se dotazovat na naše data v databázi.[20] Jedná se o Object-Relational Mapping (ORM)[19].

- **Controllery** - pomocí controllerů lze rozdělit logiku aplikace na menší celky. Controller pro daný celek obsahuje několik základních operací a zprostředkovává tak komunikaci mezi daným modelem a jeho zobrazením pro uživatele - takzvaným view.
- **View** - neboli zobrazení jsou komponenty, ve kterých specifikujeme, jak se bude konkrétní část aplikace prezentovat. View obsahují především HTML kód a data, která jim jednotlivé controllery předaly k zobrazení. Část view, kterou nástroj Laravel Schema Designer dokázal vygenerovat, byly formuláře pro jednotlivé tabulky databáze.
- **Seeding** - seeding funguje jako umělé naplnění databáze vzorovými daty. V „seedovacím“ souboru specifikujeme data, která chceme v databázi mít. Pomocí nástroje Artisan se tyto data uloží do databáze a smažou se data původní. Tato funkcionalita je užitečná především pro testovací účely.

Vygenerované části byly ovšem pouhou kostrou, specifikaci funkcí bylo třeba doplnit podle vlastních potřeb.

3.2 Imlementace MVC

Do výše vygenerovaných souborů bylo potřeba implementovat naše konkrétní *Model-View-Controller* pro pacienty, náramky a uživatele. Rozložení jednotlivých stránek vychází z wireframů v předchozí kapitole.

3.2.1 Model

Model je automaticky přiřazen k tabulce v databázi. Bylo potřeba popsat, které atributy v tabulce jsou `$fillable`, neboli vyplnitelné pomocí formuláře. Tím v Laravel zajišťuje, aby nemohly být přepsány důležité atributy, jako například `id`, které nechceme upravovat. Jedná se zároveň o bezpečnostní opatření proti útokům, aby se například v tabulce `users` neomhlo stát, že útočník přepíše atribut `is_admin` z `false` na `true` a nestane se tak administrátorem.

3.2.2 Controller

V *controllerech* byly popsány jednotlivé metody pro operace s entitami. Struktura takových metod (pro ukázku) na entitě pacientů vypadá následovně:

- `index` - zobrazí stránku(`view`) s přehledem pacientů.
- `create` - zobrazí stránku pro vytvoření nového pacienta.

- `store` - pomocí dat z vyplněného formuláře vytvoří nový záznam pacienta v databázi.
- `show` - zobrazí stránku s detailem pacienta.
- `edit` - zobrazí stránku s úpravou pacienta.
- `update` - pomocí dat z vyplněného formuláře upraví záznam pacienta v databázi.
- `destroy` - smaže daný záznam pacienta z databáze.

3.2.3 View

Teprve poté byly implementovány jednotlivé *views* zobrazující potřebný obsah. V tuto chvíli se jednalo o pouhou funkční kostru, která bude v další kapitole rozšířena a grafické styly.

Podobnou strukturu tvořilo stránka párování náramku s pacientem, kdy bylo využito již existujících modelů pacienta a náramku. Byl vytvořen samostatný controller obsahující jediné dvě metody: `index` a `update`.

3.3 Autentikace

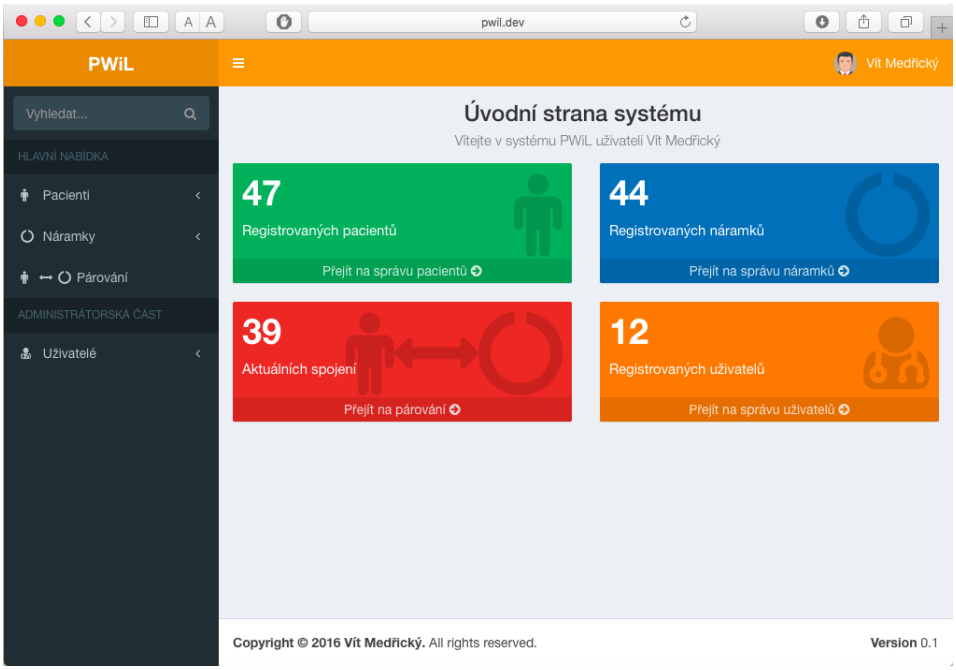
Na základě funkčních požadavků bylo v sytému potřeba implementovat přihlašování. V tomto kroku opět usnadnil práci nástroj Artisan, pomocí kterého byly vygenerovány soubory pro autentikaci. Součástí vygenerovaných souborů byly i `views` pro přihlášení, registraci a obnovení hesla, které byly upraveny podle potřeb naší aplikace. Byla odebrána možnost registrace, která v systému být nemá.[8]

3.4 Grafické styly a šablona

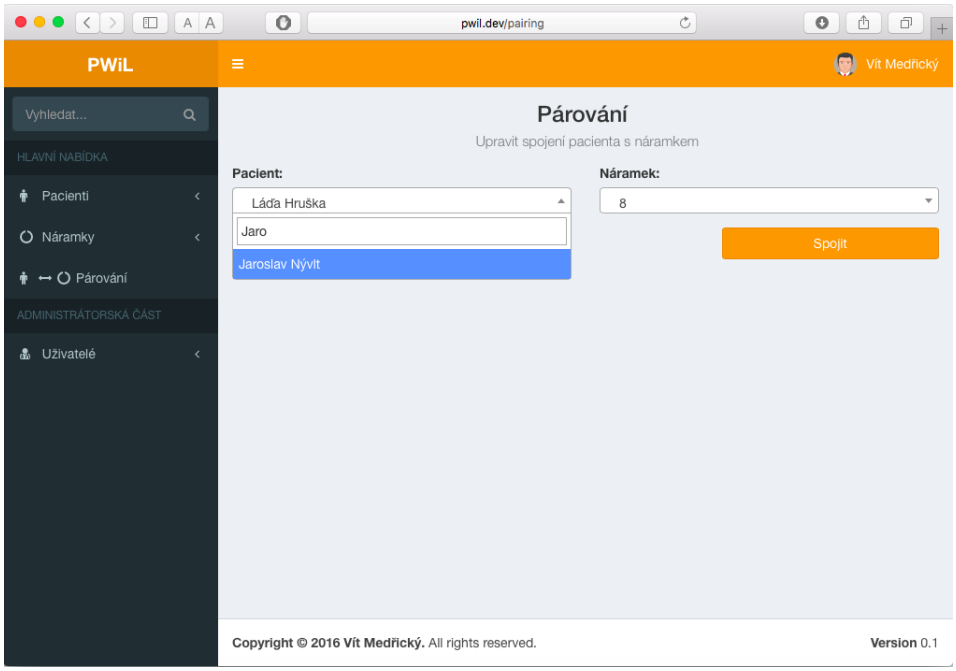
Současné uživatelské rozhraní bez grafických stylů by bylovelmi nepřehledné. Bylo tedy zapotřebí vytvořit přehledné grafické prostředí. V tomto bodě práci usnadnily předem navržené wireframy.

Požadavkem bylo stanovit jednotný vzhled a implementovat navigační menu. K tomuto účelu byla použita volně dostupná grafická šablona Admin LTE[21], která kromě menu nabízí širokou škálu ovládacích prvků, jako jsou například dlaždice na úvodní stránce (viz obr. 3.1) nebo rozbalovacího menu s vyhledáváním `select2`[22], které bylo použito na stránce s párováním (viz obr. 3.2).

[8]Tu bude zajišťovat pouze administrátor.



Obrázek 3.1: Ukázka aplikace - dlaždice a menu



Obrázek 3.2: Ukázka aplikace - rozbalovací menu select2

Pro grafickou podobu tlačítek a rozložení prvků na obrazovce byla použita knihovna Bootstrap[23], která umožňuje jednoduché úpravy vzhledu a řeší responzivní[9] design.

3.5 API

Mobilní aplikace a náramek potřebují komunikovat s backendem aplikace. Pro tyto účely bylo na základě návrhu implementováno API obsahující tyto části:

- `PatientsController` - zajišťuje odpověď mobilní aplikaci na dotaz na seznam pacientů.
- `DigestController` - zajišťuje odpověď mobilní aplikaci na aktuální změny.
- `WristbandCommunicationController` - zajišťuje příjem informací o volání od náramku.

V následujících sekcích si poslední dvě části rozebereme podrobněji.

3.5.1 DigestController

DigestController se sestavuje jednotlivé částí výpisu změn, které budou zaslány mobilní aplikaci jako odpověď na HTTP požadavek typu `GET` na adresu `/api/v1/digest`.

Tento výpis se bude skládat z následujících částí:

- `location_changes` - tato část bude obsahovat výpis pacientů, jejichž poloha se změnila od doby, kdy se mobilní aplikace ptala na tyto změny naposledy. Výpočet těchto změn provádí samostatně implementovaná třída `LocationChangesComputer`.
- `calling_for_help` - tato část výpisu bude obsahovat seznam `id` pacientů, kteří volají o pomoc.
- `patient_changes` - v této část bude pouze identifikátor, zda se změnil některý záznam o pacientech.

Jelikož jsou tyto části implementovány odděleně, je umožněno snadné přidání dalších záznamů.

[9] Při změně velikosti okna se přizpůsobují velikosti prvků na obrazovce

3.5.2 WristbandCommunicationController

Tento *controller* má na starosti komunikaci s náramkem. Jelikož nebylo doposud vybráno, jaké konkrétní náramky budou v tomto sytému použity, byl implementován controller, který očekává HTTP požadavek typu POST na adrese `api/v1/wristband-communication`. Jako posílaný parametr tohoto požadavku je očekávána MAC adresa nármaku. Controller poté zajistí, aby informace o volání byla přiřazena k danému pacientovi v databázi a mohlo tak být sestaveno volání o pomoc pro mobilní aplikaci.

3.6 Subsystém komunikující se SPoT API

V prototypu byl implementován *controller*, který zastupuje subsystém pro komunikaci s lokalizačním serverem. Tento controller umí přijmout a uložit odpověď SPoT API na poslední známé lokace pacientů (ve formátu uvedeném v dokumentaci). Pro tento subsystém ovšem nebyl implementován *scheduler*, který by zařídil cyklické dotazování. Jelikož nebylo během implementace dostupné testovací SPoT API, byla by tato funkce velmi obtížně testovatelná a proto byly upřednostněny implementace ostatních částí. Tato nedokončená část bude muset být doimplementována.

Kapitola 4

Testování

Důležitou součástí projektu je testování, pomocí kterého jsme schopni odhalit chyby vzniklé při implementaci. V této kapitole se budeme zabývat testováním zhotoveného prototypu aplikace PWiL.

4.1 Integrační testy

Bylo potřeba otestovat základní funkce, na kterých celý systém stojí. To nám umožní automatické *integrační testy*, které ověří bezchybnou komunikaci mezi jednotlivými komponentami uvnitř aplikace[25]. Laravel toto testování usnadnil již předpřipravenými třídami a testovacím nástrojem `phpunit`.

4.1.1 Test autentikace

Jelikož je v aplikaci důležité dbát na neoprávněné přístupy, byly provedeny automatické integrační testy autentikace. Testováno bylo samotné přihlášení a odhlášení a následně pokusy o přístup do systému bez přihlášení. Otestován byl také pokus o přístup do správy uživatelů bez administrátorského oprávění.

4.1.2 Test CRUD operací

Na entitách pacientů, náramků a uživatelů byly provedeny integrační testy základních *CRUD*[10] operací.

4.1.3 Test párování pacienta s náramkem

Bylo potřeba také otestovat správné propojení náramku s pacientem.

Těchto testů bylo celkem 20 a byly jimi otestovány všechny akce, které může v aplikaci dělat uživatel či administrátor.

[10] Create = přidávání, Read = zobrazení, Update = upravení, Delete = smazání

4.2 Komunikace se SPoT API

Jelikož lokalizační server Ruckus SPoT není levnou technologií, nebylo v průběhu implementace této aplikace možné zajistit vypůjčení testovacího serveru zadavatelskou firmou. Aby bylo možné otestovat reálnou komunikaci s lokalizačním serverem, bylo nutné implementovat testovací *mockup* tohoto rozhraní.

V tomto rozhraní byla imlementována funkce `last_known_locations`, která vrací poslední známe polohy pacientů ve formátu dle dokumentace.


```
▼[
  ▼{
      "floor_number": 1,
      "located_inside": true,
      "mac": "2CE6CC6D67F8",
      "timestamp": "2013-07-18T09:59:00+08:00",
      "x": 500.851874369,
      "y": 223.851874369
    },
  ▼{
      "floor_number": 1,
      "located_inside": true,
      "mac": "2CE6CC6D67F8",
      "timestamp": "2013-07-18T09:59:02+08:00",
      "x": 510.851874369,
      "y": 203.851874369
    },
  ▼{
      "floor_number": 1,
      "located_inside": true,
      "mac": "2CE6CC6D67F8",
      "timestamp": "2013-07-18T09:59:04+08:00",
      "x": 530.851874369,
      "y": 212.851874369
    }
]
```


Obrázek 4.1: Ukázka odpovědi SPoT API na lokaci pacientů

Na této verzi rozhraní bylo možné otestovat dotaz na poslední lokace pacientů a údaje z odpovědi byly úspěšně uloženy do databáze. Tento test byl proveden pouze jednorázově, ačkoliv v reálném provedení se bude aplikace dotazovat ve smyčce na stále nová data. Pro naše účely ovšem postačí otestovat, zda proběhlo správně vyřízení požadavku a data se správně uložila.

SPoT API jako takové obsahuje mnoho dalších funkcí, které nebyly implementovány, nejsou totiž pro náš prototyp podstatné.

4.3 Komunikace s náramky

Backend aplikace PWiL potřebuje komunikovat s náramky. Jelikož nebyly v průběhu implementace dostupné opravdové náramky, bylo zapotřebí provést simulaci této komunikace. Byl vytvořen test, který simuloval zaslání zprávy od náramku. Komponenta pro komunikaci s náramkem tuto informaci zpracovala a uložila do databáze. Test proběhl vpořádku a informace o stavu ohrožení byly připraveny k předání dál mobilní aplikaci.

4.4 Komunikace s mobilní aplikací

V systému bylo potřeba otestovat komunikaci s mobilním zařízením. Pro tyto účely byly vytvořeny testy simulující dotazy mobilní aplikace na výpis změn a na seznam pacientů. Test dotazu na výpis změn proběhl vpořádku.


```
{
  "calling_for_help": [
    2
  ],
  "location_changes": {
    "TESTING_MAC_ADDRESS_1": {
      "location_x": 100.85,
      "location_y": 600.39,
      "timestamp": "2013-07-18 09:59:10"
    },
    "TESTING_MAC_ADDRESS_2": {
      "location_x": 500.85,
      "location_y": 600.39,
      "timestamp": "2013-07-18 09:59:09"
    }
  },
  "patient_changes": true
}
```


Obrázek 4.2: Ukázka odpovědi - Výpis změn

U testu dotazu na seznam pacientů bylo odhaleno špatné zakódování znaků českých jmen.


```
▼[
  ▼{
      "created_at": "2016-05-15 19:18:39",
      "deleted_at": null,
      "first_name": "L\u00e1\u010fa",
      "id": 1,
      "is_calling_for_help": 0,
      "last_name": "Hru\u0161ka",
      "updated_at": "2016-05-15 19:18:39"
    },
  ▼{
      "created_at": "2016-05-15 19:18:39",
      "deleted_at": null,
      "first_name": "Bo\u017eena",
      "id": 2,
      "is_calling_for_help": 1,
      "last_name": "N\u011bmcov\u00e1",
      "updated_at": "2016-05-15 19:18:39"
    },
  ▼{
      "created_at": "2016-05-15 19:18:39",
      "deleted_at": null,
      "first_name": "Sir John",
      "id": 3,
      "is_calling_for_help": 0,
      "last_name": "Anthony",
      "updated_at": "2016-05-15 19:18:39"
    },
```


Obrázek 4.3: Ukázka odpovědi - Seznam pacientů

4.5 Shrnutí testování

Díky těmto testům byly odhaleny některé chyby, které bude třeba opravit předtím, než se na projektu bude dále pracovat. Především ty testy, které ukázaly na chyby je třeba brát za pozitivní. Jelikož zadavatel s následujícím vývojem počítá, bude potřebné testování zopakovat s opravdovými náramky, mobilní aplikací a lokalizačním serverem.

Závěr

Splnění zadání

1. **Pomocí metod softwarového inženýrství analyzujte požadavky na systém a vyberte vhodné technologie pro jeho realizaci.**

   V první kapitole byla rozebrána analýza funkčních a nefunkčních požadavků na aplikaci. Na základě funkčních požadavků byly vytvořeny případy užití systému s detailními scénáři. Jako technologie pro zhotovení byl především na základě požadavků zavavatele vybrán PHP framework Laravel, grafická šablona admin LTE a grafická knihovna Bootstrap.

2. **Nastudujte funkcionalitu SPOT API, API náramku a dalších potřebných technologií.**

   Na základě dostupné dokumentace byla nastudována funkcionalita SPoT API. Byli nastudovány dokumentace nabízených náramků. Na úplném začátku projektu byl navíc nastudován požadovaný framework Laravel, aby byl autor předem seznámen s možnostmi implementace.

3. **Na základě předchozího bodu navrhněte API poskytované mobilním zařízením.**

   V druhé kapitole byl proveden návrh API pro mobilní aplikaci, který vycházel z komunikace se zadavatelem a zhotovitelem mobilní aplikace.

4. **V návaznosti na předchozí body vypracujte návrh systému.**

   Byla navržena struktura systému, databázový model pro ukládání potřebných dat a proveden návrh komunikací mezi jednotlivými komponentami systému.

5. **Navrhněte UI, zamyslete se nad způsoby použití a jak je bude UI pokrývat.**

Za účelem dobré ovladatelnosti aplikace byl proveden kompletní návrh uživatelského rozhraní webové aplikace PWiL. Byly vytvořeny wireframy, na kterých byly jasně popsány funkcionality dílčích stránek a rozložení jednotlivých prvků.

6. **Implementujte prototyp aplikace skládající se z grafického rozhraní pro uživatele a aplikačního rozhraní pro mobilní zařízení.**

   Na základě návrhu byla vytvořena databáze se vzorovými daty, která umožnila implementaci CRUD operací nad entitami sytému a propojování náramku s pacientem. Bylo vytvořeno uživatelské rozhraní na základě wireframů a pomocí externích šablony Admin LTE bylo přidáno do systému menu. K umožnění komunikace mezi aplikací PWiL a lokalizačním serverem byla implementována komponenta pro zasílání dotazů na SPoT API. Jelikož nebyly dostupné opravdové náramky, byla vytvořena komponenta, která očekává požadavky od náramků na stanovené adrese. Tato komponenta bude muset být upravena podle konfigurace opravdového náramku. K dosažení komunikace s mobilním zařízením bylo implementováno API, na kterém se mobilní aplikace může dotazovat na seznam pacientů a výpis změn.

7. **Zhotovený prototyp podrobte vhodnému testování.**

   Na zhotoveném prototypu byly provedeny automatické integrační testy CRUD operací nad entitami, testy autentikace a test propojení s náramkem. Z důvodu nedostupnosti lokalizačního serveru bylo implementováno testovací SPoT API, na kterém bylo otestováno dotazování na lokace pacietnů naší komponentou pro komunikaci s lokalizačním serverem. Pro komunikaci s náramky byl vytvořen test simulující volání náramku. Komunikace s mobilní aplikací byla otestována simulovanými požadavky, které se dotazovali API aplikace PWiL lokace pacientů a seznam změn. Během těchto testů byly odhaleny drobné implementační chyby, které bude potřeba před budoucím vývojem opravit.

Návrh dalších funkcionalit

Systém by mohl být obohacen o další funkcionality, které by rozšiřovali jeho funkční jádro. To by zajisté přispělo k jeho konkurenceschopnosti v budoucnu. Tyto funkce z ohledu na složitost nejsou náplní implementace tohoto prototypu, jedná se pouze o navrhované rozšíření.

- **Analýza funkčnosti náramků:** Zajímavým rozšířením by mohla být monitoring náramků. Bylo by možné analyzovat jejich průměrnou výdrž batrie, životnost, poruchovost. Evidovat konkrétní náramky a jejich hardwarové a softwarové potíže, byla by tak usnadněna jejich údržba.

- **Analýza pacientů:** Se souhlasem daného pacienta by mohl být monitorován jeho dlouhodobý pohyb, ušlá vzdálenost, počet volání o pomoc, či další funcionality, které by umožnil náramek. Tato data by mohla být využita nejen jako analýza zdravotního stavu pacienta. V možnosti by bylo i poskytnutí zpětného výpisu, či mapy pohybu pacienta za poslední časový úsek.
- **Analýza funkčnosti aplikace:** V aplikaci by byly monitorovány veškeré chyby, které nastaly při jejím běhu. Bylo by tak možné analyzovat spolehlivost aplikace. Zároveň by byla lépe odhalitelna slabá místa, která je třeba vylepšit. Aplikace by tak sama posktovala zpětnou vazbu pro vývojáře.

Budoucnost projektu a zhodnocení

Tuto práci lze považovat za první uzavřenou iteraci projektu. Jelikož se jedná o zhotovení systému pro zadavatelskou firmu, bude se na projektu nadále pracovat. Budou opraveny chyby, rozšířena funkcionalita a bude se diskutovat o možných změnách, případně začlenění této práce do většího projektu. Z tohoto důvodu bylo při práci na projektu dbáno na to, aby byly jednotlivé kroky co nejkvalitnější a následné úpravy co nejmenší.

Pro autora byla tato práce velkým přínosem. Byl to první projekt takového rozsahu, díky kterému se naučil praktické dovednosti v oblasti vývoje softwarového projektu a četné znalosti webových technologií. Značně tak rošířil své znalosti nabyté při předchozím studiu a získal zkušenost s vývojem v praxi, které určitě využije jak při pokračování na vývoji aplikace PWiL, tak i v budoucnu.

Literatura

[1] Google Maps is the Most-Used Smartphone App in the World. statista.com. [online]. 4 .2013 [cit. 2016-05-11]. Dostupné z: https://www.statista.com/chart/1345/top-10-smartphone-apps-in-q2-2013/

[2] Mobile Browser vs Application Preferences. statisticbrain.com. [online]. 17. 3. 2015 [cit. 2016-05-11]. Dostupné z: http://www.statisticbrain.com/mobile-browser-vs-application-preferences/

[3] C. Fritsche, A. Klein, "On the Performance of Hybrid GPS/GSM Mobile Terminal Tracking", IEEE ICC Workshops, pp.1-5, [2010]

[4] Zhang, Da, et al. "Localization technologies for indoor human tracking."Future Information Technology (FutureTech), 2010 5th International Conference on. IEEE, [cit. 2016-05-11].

[5] CS3151BBCD RTLS Wrist Tags. two-red-cells.com. [online]. [cit. 2016-05-11]. Dostupné z: http://two-red-cells.com/external/documents/RedCell.CS3151BBCD.RC-DS-014-EN-1405.pdf

[6] Functional vs Non Functional Requirements. reqtest.com. [online]. 5. 4. .2012 [cit. 2016-05-17]. Dostupné z: http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

[7] Laravel (Projects using Symfony). symfony.com. [online]. 2016 [cit. 2016-05-16]. Dostupné z: http://symfony.com/projects/laravel

[8] Data, C. J. An introduction to database systems. Addison-Wesley publ., 1975.

[9] User Interface Design Basics. usability.gov. [online]. 12.5.2016 [cit. 2016-05-12]. Dostupné z: http://www.usability.gov/what-and-why/user-interface-design.html

[10] Principles of user interface design. bokardo.com. [online]. [cit. 2016-05-12]. Dostupné z: http://bokardo.com/principles-of-user-interface-design/

[11] What are some advantages of using the database management. quora.com. [online]. 14. 10. 2015 [cit. 2016-10-14]. Dostupné z: https://www.quora.com/What-are-some-advantages-of-using-the-database-management-system-approach-rather-than-the-file-system-approach

[12] Laravel Schema Designer. http://laravelsd.com/. [online]. 2016 [cit. 2016-13-05]. Dostupné z: http://laravelsd.com/

[13] NinjaMock. ninjamock.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: http://ninjamock.com

[14] Flowchart Marker & Online Diagram Software . lucidchart.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: https://www.lucidchart.com

[15] Balsamiq Mockups. https://balsamiq.com/. [online]. 2016 [cit. 2016-05-14]. Dostupné z: https://balsamiq.com/products/mockups/

[16] Justinmind – Prototype Faster Communicate Better. justinmind.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: http://www.justinmind.com

[17] REST matters (and you need more of it). www.pluralsight.com. [online]. 16. 4. 2015 [cit. 2016-05-16]. Dostupné z: https://www.pluralsight.com/blog/tutorials/representational-state-transfer-tips

[18] Gamma, Erich. Design patterns: elements of reusable object-oriented software. Pearson Education India, 1995.

[19] Vasiliev, Yuli. Beginning database-driven application development in Java EE: using GlassFish. Apress, 2008.

[20] Eloquent: Getting Started. laravel.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: https://laravel.com/docs/5.1/eloquent

[21] Admin LTE Prewiev. alsameedstudio.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: https://almsaeedstudio.com/preview

[22] Select2. select2.github.io. [online]. 2016 [cit. 2016-05-14]. Dostupné z: https://select2.github.io

[23] Bootstrap. getbootsrap.com. [online]. 2016 [cit. 2016-05-14]. Dostupné z: http://getbootstrap.com

[24] Ruckus SPoT Loation Services API. . [online]. 14. 6. 2014 [cit. 2016-14-06]. Dostupné z: http://a030f85c1e25003d7609-b98377aee968aad08453374eb1df3398.r40.cf2.rackcdn.com/other/ruckus_spot_location_services_api_1.3.pdf

[25] Fáze a úrovně provádění testů. testovanisoftwaru.cz/. [online]. 21. 8. 2011 [cit. 2016-05-15]. Dostupné z: http://testovanisoftwaru.cz/tag/integracni-testovani/

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF