

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

## **Bakalářská práce**

# **Návrh a implementace webové aplikace pro monitorování pacientů**

# Prohlášení

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

V Plzni dne .....

Jiří Brát

# Abstrakt

Práce se zabývá jednou z částí systému, který je vyvíjen na pracovišti Nové technologie - výzkumné centrum Západočeské univerzity v Plzni. Tento systém se zabývá monitorováním biometrických signálů pacientů v pečovatelském prostředí. Teoretická část této práce obsahuje seznámení se s PC platformou Raspberry Pi. Dále tato část práce obsahuje seznámení s technologiemi využívanými pro implementaci webových aplikací. Praktická část práce se zabývá implementací webové aplikace na platformě Raspberry Pi a simulací reálného zatížení systému.

# Abstract

The bachelor thesis deals with one part of the system, which is being developed at a workplace of The New Technologies - Research Centre University of West Bohemia. This system concerns with monitoring of biometric signals by patients in a hospital care environment. The theoretical part of that bachelor thesis contains a familiarization with PC Raspberry Pi platform. Further this theoretical part contains the familiarization with technologies used for implementation of web applications. The practical part deals with the implementation of the web application at Raspberry Pi platform and the simulation of real system load.

# Obsah

1 Úvod .....	1
2 Měření biometrických signálů, PC platforma Raspberry Pi.....	2
2.1 Měření biometrických signálů.....	2
2.2 PC platforma Raspberry Pi.....	4
3 Technologie pro realizaci webových aplikací.....	8
3.1 Technologie na straně klienta.....	8
3.2 Technologie na straně serveru .....	9
3.3 Výběr webového serveru.....	11
4 Návrh a implementace webové aplikace.....	13
4.1 Popis systému a návaznosti této práce .....	13
4.2 Specifikace požadavků .....	14
4.3 Návrh architektury aplikace .....	17
4.4 Rozvržení webové aplikace .....	18
4.5 Implementace klientské část aplikace .....	25
4.6 Implementace serverové části aplikace .....	31
5 Testování a hodnocení dosažených výsledků práce .....	33
5.1 Kompatibilita prohlížečů .....	33
5.2 Simulace zátěže reálného systému .....	34
5.3 Omezení realizovaného řešení.....	37
6 Závěr .....	38
Literatura, elektronické odkazy a zdroje.....	39
Obsah přiloženého CD .....	42

# 1 Úvod

Nové technologie - výzkumné centrum Západočeské Univerzity v Plzni vyvíjí systém pro monitorování biometrických signálů a jejich prezentaci uživatelům. Jednou ze součástí tohoto systému je webová aplikace, prezentující naměřená data uživatelům z oblasti nemocničního a pečovatelského prostředí. Návrh a implementace této aplikace je předmětem této bakalářské práce.

Teoretická část této práce je věnována seznámení s měřením biometrických signálů pacienta v pečovatelském prostředí, dále seznámení s PC platformou Raspberry Pi a technologiemi pro realizaci webových aplikací.

Praktická část práce se zabývá popisem návazností implementovaného webového portálu na ostatní části systému. Dále je v této části popsán návrh a implementace webového portálu. V poslední části praktické části je popsáno testování webového portálu a simulace reálného zatížení zařízení Raspberry Pi.

## **2 Měření biometrických signálů, PC platforma Raspberry Pi**

### **2.1 Měření biometrických signálů**

Tato část kapitoly se zabývá obecným popisem systémů pro měření biometrických signálů. Přitom jsem vycházel ze zdrojů [ 1, 2 ].

Měření biometrických signálů poskytuje ošetřovatelskému personálu okamžitou informaci o zdravotním stavu pacienta. Nejčastěji je monitorování biometrických signálů využíváno na operačních sálech pro sledování operovaného pacienta. Dále se setkáme s monitory biometrických signálů na odděleních JIP a ARO, kde tyto monitory zajišťují nepřetržité sledování osob v kritickém stavu.

Systém pro monitorování biometrických signálů se skládají z několika částí. V současné době zaznamenáváme snahu o standardizaci těchto systémů.

#### **2.1.1 Senzory**

Senzory se používají ke snímání životních funkcí člověka. V případě sledování neelektrických vlastností jsou tyto vlastnosti převedeny na elektrické během měření. Senzory nijak nezpracovávají naměřené hodnoty. Slouží pouze pro sběr dat. Naměřené hodnoty jsou přenášeny komunikačním rozhraním do řídicí jednotky. V minulosti jeden senzor sledoval jednu veličinu. V dnešní době dokáže jeden senzor souběžně sledovat několik veličin.

Mezi běžně sledované životní funkce patří tělesná teplota, dechová frekvence, srdeční tep, krevní tlak, srdeční výdej a další.

## **2.1.2 Komunikační rozhraní**

Naměřené hodnoty jsou ze senzoru do řídicí jednotky odeslány skrze komunikační rozhraní. Komunikační rozhraní může být realizováno buď kabelovým připojením, nebo některou z bezdrátových technologií.

Kabelové připojení je často použito na odděleních JIP a ARO. Mezi hlavní výhody tohoto rozhraní patří stabilita a bezpečnost. Komunikace není tolik ovlivňována okolními vlivy jako v případě bezdrátového propojení. Tento typ propojení je často použit v případě, že pacient se v průběhu měření nepohybuje.

Bezdrátové propojení senzorů s řídicí jednotkou však přináší velkou výhodu mobility. Pacient se může se senzorem volně pohybovat a není omezen pouze na délku propojovacího kabelu. S příchodem elektricky nenáročných technologií pro bezdrátovou komunikaci jako jsou Bluetooth a ZigBee, je možné senzory napájet z vestavěné baterie.

## **2.1.3 Řídicí jednotky**

Řídicí jednotka zajišťuje vyhodnocování naměřených signálů ze senzorů. Dále řídicí jednotka zajišťuje uložení signálů ve vhodném formátu. Naměřená data mohou být upravována a dopočítávána pro účely vizualizace.

## **2.1.4 Signalizační zařízení**

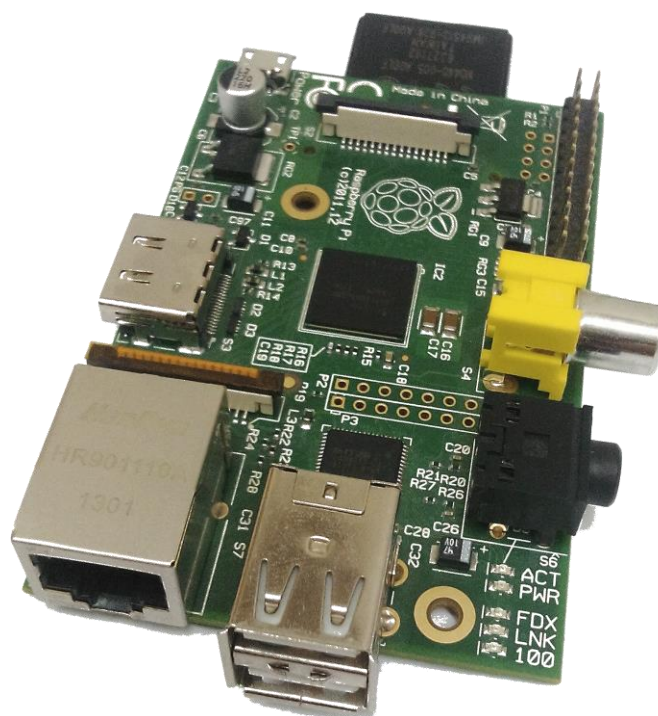
Pro účely upozornění pečovatelského personálu má velký význam signalizační zařízení, propojené s řídicí jednotkou. Upozornění může být realizováno buď vizuálně např. rozsvícením varovného světla, nebo akusticky např. spuštěním zvukového alarmu. Signalizace však může být realizována i jinými způsoby.

## 2.2 PC platforma Raspberry Pi

Počítač Raspberry Pi (viz obrázek 1) je vyvíjen nadací Raspberry Pi Foundation. V čele této nadace stojí Eben Upton, studijní ředitel pro informatiku na Cambridgeské univerzitě. Během výuky na Cambridgeské univerzitě začínal zaznamenávat u svých studentů snížené znalosti programování v porovnání s jeho generací. Rozhodl se tedy vytvořit počítač Raspberry Pi za účelem podpoření výuky. Původně Upton neměl tak ambiciózní cíl jako vytvořit světově známý počítač. Šlo mu o nadchnutí studentů. Z těchto důvodů kladl důraz na nízkou cenu, malé rozměry a celkovou univerzálnost počítače. Krátce po uvedení Raspberry Pi se stal tento mini počítač velmi oblíbeným a je díky svým vlastnostem využíván v mnoha projektech. [ 3, 4 ]

Díky své univerzálnosti Raspberry Pi zvládne cokoliv, co zvládne běžné stolní PC. Jediné, čím se bude lišit, je výkon, avšak ten v některých situacích nemusí být téměř znatelný. Raspberry Pi tak má širokou využitelnost. Raspberry Pi je možné použít např. jako řídicí jednotku robotů, jako webový server, jako kontrolér zabezpečení domácnosti, jako meteorologickou stanici, atp. [ 3, 4 ]





Obrázek 1 – Základní deska Raspberry Pi model B

## 2.2.1 Hardwarová specifika počítače Raspberry Pi model B

### Procesor a operační paměť

Srdcem Raspberry Pi je výkonný procesor Broadcom BCM2835. Jde o procesor typu SoC (system-on-chip). Jinak řečeno jedná se o jednočipový počítač, který se vyznačuje tím, že jeden integrovaný obvod v sobě zahrnuje vše potřebné pro běh celého systému. Procesor BCM2835 využívá redukovanou instrukční sadu, která se označuje jako ARM [ 3 ].

Raspberry Pi obsahuje jeden paměťový čip s kapacitou 512 MB. Tato paměť se dělí na dvě části. Jedna část je přiřazena hlavnímu procesoru a druhá část je přiřazena grafickému procesoru. To jak velká část paměti bude vyhrazena hlavnímu procesoru a jak velká část paměti grafickému procesoru je možné nastavit v konfiguračním souboru *config.txt*, který se nachází v adresáři */boot*. Toto

nastavení je klíčové pro ovlivnění výkonu počítače Raspberry Pi pro konkrétní použití. Optimalizace Raspberry Pi pro použití jako webový server je podrobněji popsáno v kapitole 2.2.2. [ 3 ]

## **Napájení**

Počítač Raspberry Pi je napájen pomocí MicroUSB konektoru s napětím 5V. Pro tyto účely je nejvhodnější použít běžný adaptér pro nabíjení mobilních telefonů. Pro napájení je ale možný použít jakýkoliv běžný USB port. Spotřeba Raspberry Pi se pohybuje kolem 3 W [ 4 ].

## **Uložiště**

Základní deska Raspberry Pi je vybavena slotem pro SD kartu. SD karta je použita jako primární uložení zařízení a systém lze nabootovat pouze z SD karty. Tento typ uložení má své výhody ale i nevýhody. Mezi hlavní výhody patří malé rozměry. Mezi hlavní nevýhody patří rychlost čtení a zápisu. Rychlost čtení a zápisu je ovlivněna typem karty [ 1 ]. Mezi další nevýhody patří velikost SD karty. Pokud je počítač Raspberry Pi použit jako domácí multimediální centrum, tak velikost SD karty pravděpodobně nebude dostatečná. Naštěstí pro tyto účely je možné připojit externí disk přes rozhraní USB. Základní deska Raspberry Pi model B je vybavena dvojicí USB portů. [ 4 ]

## **Audio/video výstup**

Pro připojení monitoru nebo televize je možno použít buď RCA phono konektor, HDMI (High Definition Multimedia Interface) konektor nebo DSI (Display Serial Interface). RCA konektor je žlutý a nachází se mezi 3,5 mm jackem a GPIO konektory. HDMI konektor je umístěn naproti RCA konektoru a jako jediný je na druhé straně desky. DSI konektor se nachází nad slotem pro SD kartu. V případě HDMI je spolu s obrazem přenášen i zvuk. V případě použití konektorů RCA a DSI se přenáší pouze obraz a pro přenos zvuku je třeba použít

3,5 mm jack, který je umístěn vedle žlutého RCA konektoru. [ 4 ]

### **Klávesnice, myš a další USB zařízení**

Jak již bylo výše zmíněno, základní deska Raspberry Pi počítače je vybavena dvojicí USB portů. Ty je možné použít pro připojení myši a klávesnice nebo dalších USB zařízení jako jsou např. WiFi adaptér, Bluetooth adaptér, další zvukové karty atp. [ 4 ]

## **2.2.2 Raspberry Pi jako webový server**

Výkonnost webového serveru ve značné míře ovlivňuje velikost použitelné paměti. Jak bylo popsáno v kapitole 2.2.1, Raspberry Pi model B disponuje 512 MB paměti. Tato paměť je sdílená mezi hlavním a grafickým procesorem. Pro běh webového serveru není potřeba, aby bylo nainstalováno grafické prostředí systému. Z tohoto důvodu není potřeba přiřazovat grafickému procesoru mnoho paměti. Naopak čím více paměti bude přiřazeno hlavnímu procesoru, tím výkonnějším bude webový server [ 1 ]. Pro účely provozu webového serveru jsem se rozhodl celkovou paměť 512 MB rozdělit na 448 MB pro hlavní procesor a 64 MB pro grafický procesor. Pro změnu velikosti paměti pro grafický procesor je nutné provést 2 kroky:

1. Do souboru */boot/config.txt* přidat řádek *gpu\_mem=64* na konec souboru a soubor uložit.
2. Restartovat Raspberry Pi.

Více informací o rozdělení paměti uvedeno ve zdroji [ 3 ].

## **3 Technologie pro realizaci webových aplikací**

### **3.1 Technologie na straně klienta**

Pro výběr vhodných kandidátů jsem použil seznam nejpoužívanějších webových technologií na straně klienta [ 6 ]. U těchto technologií jsem zjišťoval, zda splňují stanovená kritéria a zda jsou vhodné pro implementaci webové aplikace.

#### **3.1.1 Java applet**

Java applet je nejméně využívanou technologií využívanou na straně klienta. Java applet je naprogramovaný v programovacím jazyce Java, který je spuštěn uvnitř prohlížeče. Hlavní výhodou Java appletů spočívá ve srovnatelných možnostech s desktopovými aplikacemi. Tato schopnost může hrát klíčovou roli v některých specifických případech. Další předností Java appletů je jazyk, ve kterém jsou naprogramovány. Vývoj aplikací v programovacím jazyce Java je díky vlastnostem tohoto jazyka snadnější. Mezi další výhody patří množství programátorů znalých tohoto programovacího jazyka. [ 7 ]

#### **3.1.2 Adobe Flash**

Adobe Flash je procentuálně druhá nejpoužívanější technologie využívaná na straně klienta [ 8 ]. Tato technologie je často spojována s video hrami, přehráváním multimediálního obsahu a s interaktivními animacemi. Pro použití této technologie je zapotřebí použít speciální komerční grafické nástroje. Tato technologie využívá vlastní objektově orientovaný programovací jazyk ActionScript. [ 8 ]

### **3.1.3 Microsoft Silverlight**

Další technologií je Microsoft Silverlight. Jde o konkurenční technologii k dále zmíněné technologii Adobe Flash. Schopnosti této technologie jsou víceméně stejné jako v případě Adobe Flashe. K programování se používá jazyk C#. Tato technologie dobře spolupracuje s dalšími technologiemi společnosti Microsoft založenými na .NET. [ 9 ]

### **3.1.4 JavaScript – zvolená technologie**

JavaScript je jazyk, který byl vytvořen pro oživení statických HTML stránek. Postupem času se zdokonaloval a vyvíjel a tím se také výrazně měnil. JavaScript umožňuje široké využití webových aplikací. JavaScript umožňuje komunikovat s webovým serverem bez nutnosti opětovného načtení webové stránky ze serveru. Tato technologie je označovaná jako AJAX. JavaScript měl už od počátku velké možnosti v oblasti ovládání obsahu webové stránky. Dnes je možné v rámci webové stránky měnit téměř celý její obsah. Je možné v průběhu prohlížení přidávat, mazat a upravovat obsah, upravovat styly zobrazení jednotlivých částí a mnoho dalšího. Všechny tyto schopnosti dělají z JavaScriptu velmi silný nástroj. JavaScript umožňuje programovat webové aplikace běžící ve webovém prohlížeči. Tyto webové aplikace se označují jako SPA (Single Page Applications). [ 10 ]

## **3.2 Technologie na straně serveru**

Server je v tomto případě realizován zařízením Raspberry PI. Jedná se o zařízení, které není tak výkonné jako běžné servery. Na tento důležitý fakt bylo třeba brát zřetel při výběru technologií, které budou provozovány na tomto zařízení. [ 11 ]

### **3.2.1 ASP.NET**

Vychází z technologie ASP (Active Server Pages). Jde o technologii pro tvorbu webových aplikací. Hlavním konkurentem je JSP – viz níže. Je možné použít jakýkoliv jazyk podporující CLR. Oproti PHP je aplikace předkompilována, to znamená, že se provádí o něco rychleji než PHP. Hlavní nevýhodou však je, že ASP.NET je primárně určeno pro běh za pomoci Microsoft Internet Information Services (IIS). Jedná se o proprietární server společnosti Microsoft, který je publikován jako součást operačního systému Microsoft Windows. [ 12 ]

### **3.2.2 JavaServer Pages**

JavaServer Pages (JSP) stejně jako dále popisované PHP a výše uvedené ASP.NET je technologie pro tvorbu dynamických HTML stránek. Zdrojový kód HTML se kombinuje s jazykem Java. JSP spolupracuje s Apache Tomcat serverem. Dále je pro běh nutné běhové prostředí JVM. [ 13 ]

### **3.2.3 PHP – zvolená technologie**

PHP [ 14 ] je skriptovací jazyk, který slouží k tvorbě dynamických HTML stránek. PHP je platformě nezávislý. Nejčastěji je provozován na operačním systému Linux. Jde o nejrozšířenější programovací jazyk dynamických webových stránek. Je tedy vhodným kandidátem pro použití na zařízení Raspberry Pi.

Já jsem se rozhodl pro programovací jazyk PHP. Důvodů pro to bylo několik. PHP je možné integrovat do většiny webových serverů. PHP nevyžaduje běhové prostředí jako v případě JSP. Dalším důvodem byly zkušenosti s programováním v tomto jazyce.

## PHP Slim Framework

Vyhledáváním na internetu jsem došel k Slim frameworku <sup>1</sup>. Jedná se o PHP micro framework, který usnadňuje programování malých webových aplikací a API. Funkčnost frameworku zajišťuje jen několik málo tříd, to znamená, že jde o malý framework. Dokumentace je zpracovaná dobře. Tento framework jsem vyhodnotil jako vhodný k použití. Vybraný Framework jsem porovnal s několika dalšími. Zjistil jsem, že neposkytují žádné významně odlišné vlastnosti než Slim framework. Rozhodl jsem se proto použít tento framework pro serverovou část aplikace. Slim Framework používá architekturu REST [ 15 ].

Rozhraní REST se používá pro přístup ke zdrojům. Každý zdroj lze identifikovat pomocí a URI. REST dále definuje čtyři přístupy k těmto zdrojům (GET, POST, PUT, DELETE)[ 15 ]. Slim Framework pro tyto účely poskytuje čtyři funkce: *get*, *post*, *put*, *delete*. Tyto funkce odpovídají metodám přístupu, jak je definuje REST. Každá z těchto funkcí má tři parametry. Prvním z nich je URI. URI slouží k jednoznačné identifikaci zdroje. Druhým parametrem je funkce, která zajišťuje autorizaci požadavku. Tímto způsobem je zajištěno, že uživatel může provádět pouze ty operace, na která má oprávnění. Třetím parametrem je název obslužné funkce.

### 3.3 Výběr webového serveru

Každá webová aplikace potřebuje ke svému běhu webový server. Webových serverů je na výběr několik. Každý z webových serverů obsahuje, kromě společných vlastností, specifické funkce, které jsou významné pro různá programová řešení. Já jsem se zajímal o to, jaký webový server bude nejvhodnější pro běh na zařízení Raspberry Pi. Tímto tématem se zabýval Jeremy Morgan na svém blogu [ 16 ]. Provedl komplexní test čtyř nejběžnějších webových serverů za účelem nalezení vhodného serveru pro PC platformu Raspberry Pi. V testu se

---

<sup>1</sup><http://www.slimframework.com>

nejlépe umístil server NGINX<sup>2</sup>. Z tohoto důvodu jsem se ho rozhodl použít pro účely této práce.

---

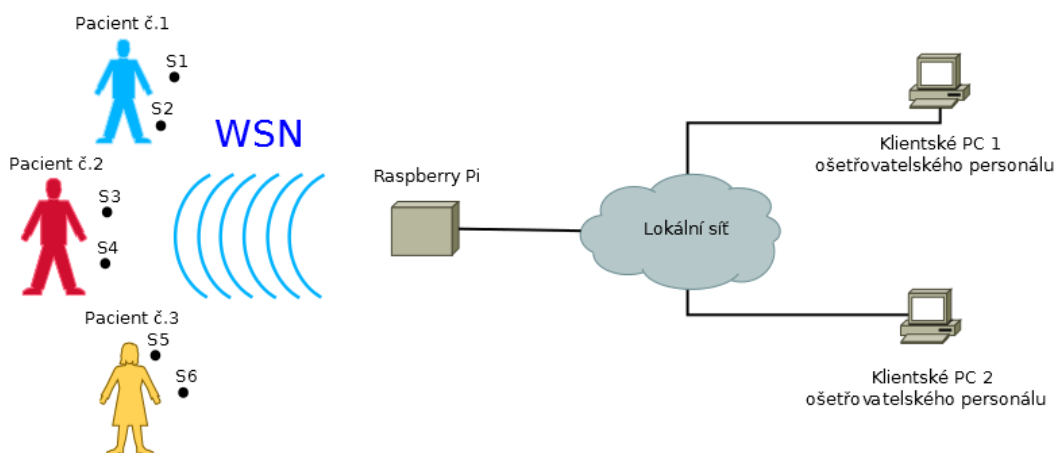
<sup>2</sup> <http://nginx.com>



# 4 Návrh a implementace webové aplikace

## 4.1 Popis systému a návaznosti této práce

Webová aplikace, jejíž návrh a vývoj je předmětem této práce, je součástí systému pro monitorování biometrických signálů pacientů. Celé řešení tvoří bezdrátové senzory, které snímají teplotu, řídicí jednotka Raspberry Pi, která zajišťuje shromažďování a zpracování naměřených dat, a klientské počítače ošetřovatelského personálu, které skrze webový prohlížeč přistupují k informacím z řídicího počítače (viz Obrázek 2).



Obrázek 2 – Výsledná architektura systému

Webová aplikace se bude skládat ze serverové části a klientské části. Serverová část přijímá data z existující části systému, která provádí sběr dat ze senzorů biometrických signálů. Tato data budou poskytována dvěma způsoby, jedním jsou data v databázi a druhým jsou binární soubory obsahující naměřená data. Struktura databázových a datových souborů je dále v této kapitole. Serverová část webové aplikace bude poskytovat data klientské části webové aplikace, kde se data vhodným způsobem budou prezentovat uživateli. Zároveň klientská část webové aplikace bude komunikovat se serverovou částí aplikace a měnit data pacientů.

### 4.1.1 Použití databázových souborů

Pro výměnu dat mezi jednotlivými částmi systému slouží databáze. Data posledního provedeného měření jsou ukládána do SQLite<sup>3</sup> databáze. Tato data mohou být využita pro rychlý přístup k aktuálnímu stavu senzorů.

### 4.1.2 Popis datových souborů

Historická data měření se ukládají do datových souborů. Pro každý sledovaný senzor existuje samostatný binární soubor. Z důvodu minimalizace přenášených dat v síti jsou data ukládána tak, že pro každé měření jsou do příslušného souboru ukládány pouze dvě informace, a to počet vteřin od půlnoci aktuálního dne a naměřená teplota. Tyto soubory jsou organizovány pro snadnou identifikaci podle data za pomoci adresářové struktury. Struktura je tvořena datem a číslem senzoru. Například datový soubor *00012.dat* v adresářové struktuře */data/2015/02/15/00012/00012.dat* obsahuje data naměřená od půlnoci 15. února 2015.

## 4.2 Specifikace požadavků

Aplikace by měla být navržena tak, aby vyhovovala použití v nemocničním a pečovatelském prostředí, kde uživatelem bude lékařský a ošetrovatelský personál. Aplikace bude zpracovávat data o teplotě z teplotních senzorů a bude navržena tak, aby v budoucnu umožnila přidat další snímané signály bez zásadních změn ve struktuře aplikace. Aplikace umožní administraci, tj. založení, úpravu a ukončení měření pacienta. Při založení, nebo úpravě pacienta budou pacientovi přiřazeny senzory biometrických signálů. Aplikace bude prezentovat aktuální data o teplotě z teplotních senzorů pro jednotlivé pacienty a poskytovat vizuální upozornění u pacientů, u kterých naměřená teplota překročí definovanou mez. Zároveň aplikace umožní procházení historie naměřených dat pro jednotlivé

---

<sup>3</sup> <https://www.sqlite.org>

pacienty. Uživatelé aplikace budou typu uživatel, nebo administrátor. Uživatelé typu administrátor budou mít přístup k administraci uživatelů. Základním zabezpečením aplikace bude přihlašování uživatelů. Data bude aplikace přejímat z již existující části systému.

## **4.2.1 Funkční požadavky**

### **Hlavní funkce systému**

1. úvodní přihlašovací obrazovka, poskytuje základní informace o aplikaci a umožňuje přihlášení do aplikace
2. hlavní obrazovka, zobrazuje aktuální, dynamicky se aktualizující data o pacientech v systému, včetně zvýraznění pacientů, u kterých byly překročeny limity naměřených hodnot, nebo jinak vyhodnoceny mimořádné stavy, umožňuje přechod na prohlížení historie měření pacienta, ukončení měření (smazání) pacienta (pro uživatele s potřebným oprávněním, smazání pacienta neodstraňuje jeho datové záznamy ze systému, ale pouze ukončuje jeho měření, pacient je stále viditelný v historii pacientů) a editaci pacienta (editace údajů a změna asociovaných senzorů)
3. historie měření pacienta, umožňuje prohlížet celou historii měření pro pacienta, u kterého měření stále probíhá, nebo již bylo ukončeno a pacient byl vymazán ze seznamu aktivních pacientů
4. historie pacientů, zobrazuje seznam všech pacientů, kteří byli v systému zavedeni, a jejich měření probíhá, nebo již bylo ukončeno vymazáním pacienta ze systému, pro každého pacienta umožňuje přechod na jeho historii měření, umožňuje úplné vymazání pacienta ze systému (pouze pro uživatele s třídou administrátor)

5. založení pacienta, umožňuje založit nového pacienta včetně asociace teplotních senzorů
6. změna pacienta, umožňuje editaci údajů a změnu asociovaných senzorů

### **Třídy uživatelů**

Aplikace bude mít dvě třídy uživatelů. Aplikace by měla být navržena tak, aby bylo možné rozšíření počtu tříd bez zásadních změn ve struktuře aplikace. Třídy uživatelů jsou „uživatel“ a „administrátor“.

### **Popis práv jednotlivých tříd uživatelů**

- **uživatel** - má přístup k naměřeným datům, může zakládat a mazat pacienty
- **administrátor** – stejná práva jako uživatel, navíc může zakládat a mazat uživatele systému

### **Uložiště dat**

Webová aplikace bude přebírat data poskytnutá ostatními částmi systému. Veškerá data aplikace budou uložena v interní paměti zařízení Raspberry Pi viz kapitoly 4.1.1 a 4.1.2.

## **4.2.2 Požadavky na provozní prostředí**

Systém bude provozován na zařízení s omezenými prostředky (výpočetní výkon, paměť, rychlost přístupu na disk). Cílovou platformou je zařízení Raspberry Pi model B s operačním systémem Linux. Mezi základní parametry této platformy patří procesor BCM2835 na frekvenci 700MHz, paměť RAM o kapacitě 512MB a ethernetové rozhraní.

### **4.2.3 Omezení návrhu a implementace**

Návrh aplikace by měl zohlednit možnosti platformy, na které systém bude provozován. Cílem návrhu je minimalizace výpočetního zatížení systému a využití prostředků. Dalším požadavkem je přehlednost struktury aplikace a možná rozšíření aplikace v budoucnu. Zároveň je nutné technologie volit s ohledem na potřebu vytvoření přehledného uživatelského rozhraní a na složitější komponenty uživatelského rozhraní jako jsou tabulky a grafy. Aplikace by měla být kompatibilní s běžnými zařízeními (PC, tablet) a běžnými webovými prohlížeči (Mozilla Firefox, Google Chrome, Internet Explorer).

### **4.2.4 Uživatelská dokumentace**

Uživatelská dokumentace není vyžadována, uživatelské rozhraní by mělo být maximálně intuitivní, případně poskytovat nápovědu vhodným způsobem jako součást uživatelského rozhraní aplikace.

### **4.2.5 Ostatní požadavky**

Systém by měl být schopen obsloužit 20 uživatelů najednou s dobou odezvy do 500ms. Otestování by mělo být provedeno pomocí vhodného nástroje pro testování zatížení. Zároveň by měla být vypracována analýza vlivu počtu současných požadavků na dobu odezvy systému pro různé akce.

## **4.3 Návrh architektury aplikace**

Při návrhu architektury aplikace jsem se držel klasické dvouvrstvé architektury klient-server. Na této architektuře je založena převážná většina síťových technologií. Tato architektura v případě síťových technologií předpokládá, že server je výkonnější než klient a proto vykonává výpočetně náročné operace. S tímto ohledem je většina webových technologií navrhována. Výjimkou není ani webový server. V běžných situacích, po přijetí požadavku od

klienta (webového prohlížeče), provádí webový server veškerou aplikační logiku a výkonnostně náročné operace. Mezi tyto operace patří např. práce se soubory, práce s databázovým serverem apod. Výsledkem jeho činnosti je pak řetězec znaků, který je odeslán zpět klientovi. Klient, v tomto případě webový prohlížeč, pouze zobrazí výsledek uživateli. Raspberry Pi ovšem není primárně vyvíjen pouze pro použití jako webový server. Pro ukládání dat je použita paměťová karta. Čtení a zápis na paměťovou kartu nejsou tak rychlé jako v případě moderních SSD disků. Z těchto důvodů jsem přemýšlel nad tím, jak optimalizovat architekturu za účelem snížení zatížení zařízení Raspberry Pi. Rozhodl jsem se přesunout aplikační logiku ze serveru na klienta. Tento požadavek jsem bral v úvahu při výběru technologií pro implementaci aplikace.

## 4.4 Rozvržení webové aplikace

Před započítím implementace webové aplikace jsem provedl rozvržení. Snažil jsem se aplikaci udělat co nejpřehlednější, proto jsem se při návrhu snažil eliminovat počet kroků, které je nutno provést pro získání potřebné informace. Na základě toho jsem se rozhodl aplikaci rozdělit do šesti částí:

- přihlášení do aplikace
- hlavní stránka
- nový pacient
- detail pacienta
- historie pacientů
- informace o senzorech

Mimo těchto částí ještě aplikace obsahuje horní panel, který je zobrazen ve všech částech aplikace. Panel obsahuje seznam tlačítek s názvy částí, které jsou

v daném okamžiku uživateli dostupné. Pro přechod do jiné části aplikace je tedy možno vždy použít jedno z tlačítek ze seznamu. Pokud se uživatel nachází v některé části aplikace, je vždy zvýrazněno příslušné tlačítko s názvem části. Panel obsahuje také tlačítko pro odhlášení. Tento přístup usnadňuje uživateli navigaci v aplikaci a poskytuje mu vždy jasnou představu o tom, v jaké části se právě nachází (viz obrázek 3).



*Obrázek 3: Navigační panel*

#### 4.4.1 Přihlášení do aplikace

Z bezpečnostních důvodů aplikace vyžaduje přihlášení uživatele před začátkem používání aplikace. Přihlášení bude realizováno jednoduchým přihlašovacím dialogem (viz obrázek 4).

Dialog bude obsahovat dvě vstupní pole, aby mohl uživatel zadat **uživatelské jméno** a **heslo**. Dále bude formulář obsahovat tlačítko **Přihlásit se** pro potvrzení údajů a provedení autentizace uživatele.

Při zobrazení přihlašovacího dialogu bude navigační panel obsahovat místo seznamu tlačítek s jednotlivými částmi aplikace pouze jediné tlačítko **Přihlásit se**.

BioMon Portál

Přihlásit se

Přihlašovací jméno: login

Heslo: heslo

Přihlásit se

Obrázek 4: Přihlašovací dialog

## 4.4.2 Hlavní stránka

Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku. Hlavní stránka uživateli poskytuje přehledné zobrazení stavu sledovaných senzorů a vývoj teploty jednotlivých senzorů v čase. K přehlednému zobrazení těchto informací jsem se rozhodl hlavní stránku rozdělit na dvě části.

V první části hlavní stránky bude umístěna tabulka, která poskytuje přehled o sledovaných pacientech a aktuálních stavech senzorů. Tabulka bude realizována komponentou *DataTables*. Bude obsahovat čtyři sloupce. V prvním sloupci bude název pacienta, druhý sloupec bude obsahovat teploty na jednotlivých senzorech daného pacienta, třetí bude obsahovat identifikaci senzoru (číslo a popis). Ve čtvrtém sloupci budou umístěna obslužná tlačítka pro vyvolání detailu nebo odstranění příslušného pacienta.

Druhá část stránky bude obsahovat graf s vývojem teplot aktuálně zobrazených pacientů. Graf bude realizován komponentou *dygraphs*<sup>4</sup>. Aby bylo

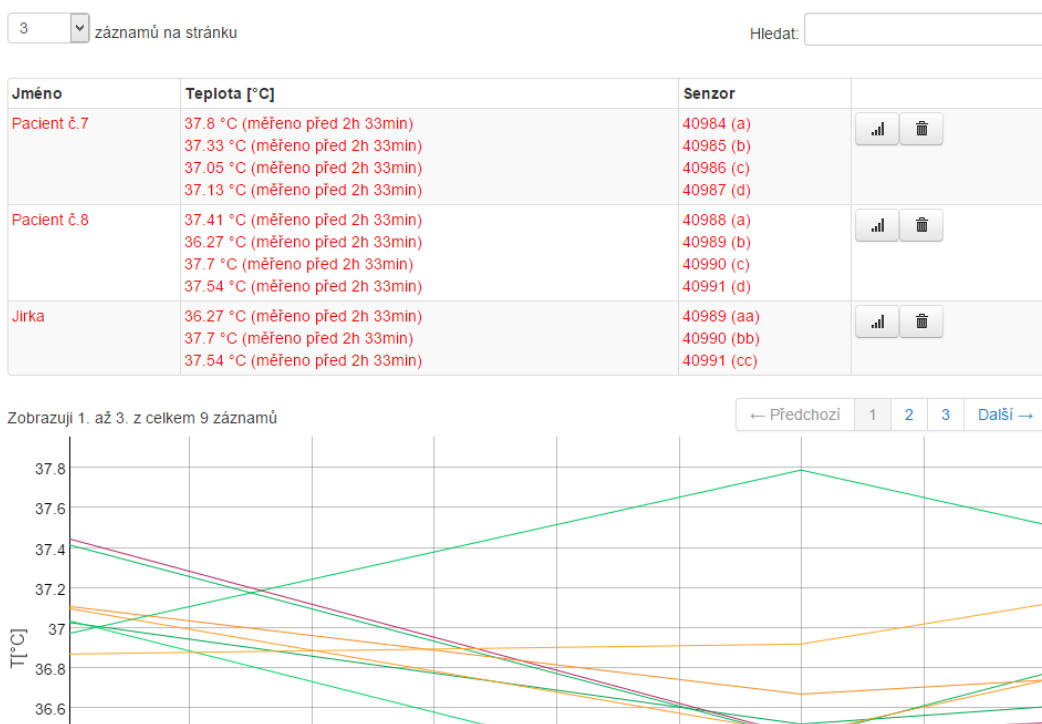
---

<sup>4</sup> <http://dygraphs.com>



možné sledovat vývoj teploty, bude graf zobrazovat vývoj teploty v aktuálním dni. Každý senzor bude vykreslen jinou barvou, aby bylo možné senzory od sebe v grafu odlišit (viz obrázek 5).

Tabulka a graf budou vždy zobrazovat stejné hodnoty. Pokud dojde ke změně stavu tabulky (změna počtu záznamů na stránku, změna řazení sloupce nebo přechod na jinou stránku tabulky), dojde k překreslení grafu pro zobrazení aktuálních dat.



Obrázek 5: Hlavní obrazovka

### 4.4.3 Nový pacient

Pro vytvoření nového pacienta bude sloužit dialogové okno s formulářem (vizte obrázek 6). Dialogové okno bude zobrazeno po vybrání tlačítka **Nový pacient** v navigačním panelu.

Formulář bude obsahovat vstupní pole pro zadání jména pacienta, tabulku

se seznamem aktivovaných senzorů a ovládací tlačítka **Storno** a **Vytvořit pacienta**. Ke každému senzoru bude možné zadat jeho popis pro snazší identifikaci. To je vhodné především v případě, že bude pacientovi přiřazeno více senzorů.

Po zadání všech potřebných údajů uživatel stiskne tlačítko **Vytvořit pacienta**. Tím dojde k uložení údajů a uživatel bude přesměrován na **Hlavní stránku**. Pokud se uživatel rozhodne zrušit proces vytváření nového pacienta, stiskne tlačítko **Storno** a dojde k přesměrování na úvodní obrazovku.

**Nový pacient** ×

Jméno pacienta

*Položte vybraný senzor na libovolnou vodorovnou plochu, např. stůl. Nechte senzor ležet alespoň 5 vteřin, pak senzor obraťte a položte zpět na stůl. Takto označený senzor se během několika vteřin objeví v tomto okně. Při použití více senzorů pro jednoho pacienta lze vyplnit stručný popis senzoru pro přehlednější interpretaci naměřených teplot.*

ID senzoru	Popis senzoru
40963	<input type="text"/>

Zrušit **Vytvořit pacienta**

Obrázek 6: Dialog pro vytvoření nového pacienta

#### 4.4.4 Detail pacienta

Pokud bude uživatel chtít zobrazit vývoj teplot pouze pro jednoho pacienta, stiskne tlačítko **Historie měření** u vybraného pacienta na hlavní stránce. Tím bude uživateli zobrazen detailní stránka pacienta. Tato stránka slouží pro vizualizaci

naměřených dat pro jednoho konkrétního pacienta. Stránka obsahuje graf pro zobrazení naměřených hodnot ze senzorů a kalendář.

Graf bude reprezentován komponentou *dygraphs* stejně jako na hlavní stránce, avšak v tomto případě bude zobrazovat pouze data ze senzorů pro vybraného pacienta. To přispěje k lepší orientaci v grafu obzvláště v případě, kdy budou mít pacienti každý více senzorů.

V případě, že bude chtít uživatel zobrazit vývoj teplot z předešlých dní, bude možné použít kalendář. Kalendář bude reprezentován komponentou *jQuery DatePicker*<sup>5</sup> (viz obrázek 7). Kalendář bude obsahovat tlačítka pro dny, ve kterých došlo k měření hodnot senzorů. Po stisknutí tlačítka vybraného dne, dojde k překreslení grafu.



Obrázek 7: Dialog pro výběr data

<sup>5</sup> <https://jqueryui.com/datepicker/>










## 4.4.5 Historie pacientů

Na hlavní stránce jsou zobrazeny pouze informace o aktuálně sledovaných pacientech. Pokud bude uživatel vyžadovat zobrazení naměřených dat pro pacienty, u kterých již bylo měření ukončeno, použije sekci **Historie** pacientů. Tato sekce obsahuje tabulku, ve které je možné najít všechny pacienty. U pacientů je uveden datum zahájení měření, ukončení měření a počet aktuálně přiřazených senzorů. Je tedy možné i zpětně dohledat pacienty, u kterých probíhalo měření v minulosti.

Tabulka je reprezentována komponentou *DataTables* (viz obrázek 8). Ta zajišťuje snadné a rychlé nalezení hledaného pacienta i v případě, že je v systému evidováno velké množství pacientů.

**Historie pacientů**

10 ▼ záznamů na stránku Hledat:

Jméno	Zahájení měření	Konec měření	Počet senzorů	
Pacient č.1	po 11.05.2015 15:42	-	4	
Pacient č.2	po 11.05.2015 15:42	-	4	
Pacient č.3	po 11.05.2015 15:42	-	4	
Pacient č.4	po 11.05.2015 15:42	-	4	
Pacient č.5	po 11.05.2015 15:42	-	4	
Pacient č.6	po 11.05.2015 15:42	-	4	
Pacient č.7	po 11.05.2015 15:42	-	4	
Pacient č.8	po 11.05.2015 15:42	-	4	
Pacient č.9	po 11.05.2015 15:42	st 13.05.2015 07:20	4	

Obrázek 8: Historie pacientů

## 4.4.6 Informace o senzorech

Pro zjištění aktuálního stavu čidel bude sloužit sekce **Senzory**. Ta umožňuje zobrazení aktuálních stavů jednotlivých čidel - nejen teploty, ale i stav baterie, sílu signálu atd. Tyto informace jsou užitečné pro správce systému například v případě nefunkčnosti čidla. Je zde možné zjistit důvod nefunkčnosti

čidla, tedy zda je zapříčiněna špatným signálem, vybitou baterií aj. Tato stránka byla vytvořena pro potřeby vývoje a je možné, že bude v budoucnu odstraněna.

Informace o čidlech jsou přehledně zobrazeny v tabulce, která je reprezentována komponentou *DataTables* (viz obrázek 9).

#### Informace o senzorech

25  záznamů na stránku Hledat:

#	Napětí baterie	Teplota	Acc	RX total/RX buffered	RX power	Čas posledního příjmu	Meas idx	Identified
40999	0.0V	37.78°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40998	0.0V	36.72°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40997	0.0V	37.15°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40996	0.0V	36.85°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40995	0.0V	36.7°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40994	0.0V	36.76°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 2s	0	0
40993	0.0V	36.52°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40992	0.0V	37.13°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40991	0.0V	36.34°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40990	0.0V	36.75°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40989	0.0V	36.94°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40988	0.0V	37.76°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0
40987	0.0V	36.69°C	0.0 / 0.0 / 0.0	0 / 0	0	před 2h 33m 3s	0	0

Obrázek 9: Informace o senzorech

## 4.5 Implementace klientské část aplikace

### 4.5.1 Proces přihlášení do aplikace

Vstupním bodem aplikace je soubor *index.php*. Tento soubor nejprve načte konfigurační soubor *config.php* a následně *checklogin.php*. V souboru *checklogin.php* se provede ověření, zda je uživatel přihlášený do systému. Ověření probíhá tak, že pokud existuje session, která lze identifikovat na základě cookie, je uživatel přihlášen. V tomto případě pokračuje provádění kódu v souboru *index.php* a uživateli je zobrazena hlavní stránka aplikace. Pokud uživatel přihlášen není, je přesměrován zpět na přihlašovací dialog a požádán o kontrolu přihlašovacích údajů.

Tímto mechanismem je aplikace chráněna proti neoprávněnému přístupu. Při ověřování přihlašovacích údajů jsou také v případě úspěšného přihlášení načtena oprávnění přihlašovaného uživatele. Na základě oprávnění je potom přizpůsobeno rozhraní pro běžného uživatele nebo pro administrátora.

Přihlašovací dialog je realizován z bezpečnostních důvodů separátně v souboru *login.php*. V tomto souboru je vytvořen formulář podle návrhu popsaného v této kapitole. Na tento dialog je uživatel přesměrován v případě, že není přihlášen. Po odeslání přihlašovacích údajů je řízení předáno souboru *index.php*, který provede přihlášení.

## 4.5.2 Proces inicializace aplikace

Pokud je proces přihlášení úspěšný, v souboru *index.php* dojde k načtení potřebných stylů a JavaScriptového kódu aplikace. Soubor *index.php* také obsahuje šablony pro všechny sekce aplikace. Tím, že je celá aplikace najednou načtena do webového prohlížeče klienta, se sníží počet http dotazů v průběhu používání aplikace a tím je i sníženo množství zátěže na zařízení Raspberry Pi.

Po načtení potřebných knihoven dochází nejprve k načtení konfiguračního souboru *config.js* klientské části aplikace. V tomto souboru se nachází konfigurace aplikace, jako například: jak často aktualizovat hodnoty v senzorů na hlavní stránce, při jaké teplotě má být signalizované varování o teplotě a podobně.

## 4.5.3 Zajištění dostatečné rozšiřitelnosti

K zajištění dostatečné rozšiřitelnosti aplikace jsem se rozhodl vnořit několik *Backbone.View* do sebe. Jelikož skládáním objektů *Backbone.View* do složitějších celků některé objekty plní pouze roli struktury pro další vnořené objekty, rozhodl jsem se tyto objekty pojmenovat *Layout*. V rámci jednoho *Layoutu* jsem pak definoval tzv. *regiony*, což je označení pro místa, kam je možné umístit vnořené objekty.

Na nejvyšší úrovni je *Biomon.AppLayout*. Tento layout rozděluje stránku na dvě části:

- **header** – v tomto regionu je umístěn objekt *Biomon.HeaderView*, který reprezentuje navigační panel popsáný v kapitole 4.4.
- **main** – reprezentuje zbytek plochy aplikace, což je místo, kde se vykreslují jednotlivé sekce aplikace. Tento region se při přechodu mezi sekcemi vždy překreslí vnořeným layoutem dané sekce.

Tímto rozdělením je docíleno konzistentního zobrazení napříč celou aplikací. Uživatel se může spolehnout na umístění navigačního panelu v hodní části aplikace.

#### 4.5.4 Hlavní stránka

Hlavní stránka se skládá ze dvou komponent tabulky a grafu. Každá z těchto komponent má vlastní *Backbone.View* objekt. Pro správné zobrazení v hlavní části aplikace slouží objekt *Biomon.HomeLayoutView*. Tento layout má dva regiony: *table* a *graph*. Region *table* obsahuje objekt *Biomon.HomeTableView* a region *graph* obsahuje objekt *Biomon.HomeGraphView*.

##### **Biomon.HomeTableView**

Objekt *Biomon.HomeTableView* reprezentuje vizuální část komponenty *DataTables* na hlavní stránce. Potřebná data pro vykreslení komponenty poskytuje objekt *Biomon.PatientCollection*. Tento objekt nejprve provede volání API za účelem získání seznamu sledovaných pacientů a následně z jednotlivých záznamů vytvoří objekty *Biomon.PatientModel*, které reprezentují pacienty v systému. Po úspěšném vytvoření kolekce objektů pacientů je tato kolekce předána objektu *Biomon.HomeTableView*, který za pomoci knihovny *DataTables* vykreslí

komponentu tabulky. Data v tabulce jsou automaticky aktualizována jednou za 5 sekund.

## **Biomon.HomeGraphView**

Objekt *Biomon.HomeGraphView* obstarává vykreslení grafu průběhů teplot na hlavní stránce. Data pro vykreslení zajišťuje objekt *Biomon.SensorModel*. Ten je propojený s objektem *Biomon.PatientModel*. Při aktualizaci seznamu senzorů *Biomon.PatientModel* vytvoří pro každý senzor objekt *Biomon.SensorModel* a nad tímto objektem zavolá funkci *updateSensorData*. Ta na základě čísla senzoru a aktuálního data stáhne binární soubor obsahující naměřená data. Aby bylo možné data zobrazit, je nejprve třeba data správně načíst z binární podoby. Za pomoci základních bitových operací ve funkci *bitOperations* se z binární podoby vytvoří dvourozměrné pole, které obsahuje hodnoty: počet sekund od půlnoci a teplotu ve stupních celsia s přesností na jedno desetinné místo. Tento formát odpovídá požadavkům knihovny *dygraphs* k zajištění správné interpretace.

### **4.5.5 Nový pacient**

Pro zobrazení dialogového okna s formulářem pro vytvoření nového pacienta jsem se rozhodl, že nebudu používat *Backbone.View*, abych zbytečně nekomplikovat zdrojový kód. Funkčnost zobrazení a skrytí zajišťuje sám *Twitter Bootstrap*<sup>6</sup>. Není tedy proto důvod hledat jiné řešení.

Po kliknutí na tlačítko **Nový pacient** je v souboru *header.js* zavolána funkce *navPatientCreate*. Ta nejdříve zobrazí dialogové okno pro vytvoření nového pacienta a následně v intervalu jedné sekundy volá API, aby zjistila, který ze senzorů byl aktivován a má být přiřazen novému pacientovi. Poté, co je senzor aktivován, je při dalším volání API vráceno číslo senzoru. Na základě této akce

---

<sup>6</sup> <http://getbootstrap.com/>



je senzor automaticky přidán do seznamu přiřazených senzorů. Ke každému senzoru se současně vytvoří vstupní pole, do kterého je možné zadat popis.

Po zadání jména pacienta případně popisu senzorů a potvrzením vytvoření stisknutím tlačítka **Vytvořit pacienta** dojde nejprve k vytvoření objektu *Biomon.SensorModel* pro každý z přiřazených senzorů a následně k vytvoření samotného modelu pacienta prostřednictvím objektu *Biomon.PatientModel*. Nad nově vytvořeným objektem modelu pacienta je zavolána funkce *save*, která vyvolá požadavek na uložení nového pacienta prostřednictvím API. Po úspěšném uložení modelu je model přidán do kolekce *Biomon.PatientCollection*. Tato kolekce je označena za změněnou. Na změnu kolekce pacientů může reagovat komponenta tabulky na úvodní stránce a na základě toho překreslit svůj obsah.

#### 4.5.6 Detail pacienta

Sekce detailu pacienta se skládá ze dvou komponent. První je graf průběhů naměřených hodnot a druhou komponentou je kalendář. Stejně jako na hlavní stránce obě komponenty mají vlastní *Backbone.View* objekt s názvem *Biomon.DetailGraphView* a *Biomon.DetailCalendarView*. Pro přehledné uspořádání komponent v této sekci je stejně jako na úvodní stránce využit vnitřní layout *Biomon.DetailLayoutView*.

Pokud není specifikován datum, jsou v grafu zobrazeny naměřené hodnoty pro aktuální den. Pro procházení naměřených hodnot z předešlých dní slouží komponenta kalendáře. Komponenta na základě informací o začátku a konci měření určí interval dní, pro které je možné zobrazit naměřené hodnoty v grafu:

- Pokud již bylo měření ukončeno, bere se rozdíl mezi datem začátku měření a datem ukončení měření
- V případě probíhajícího měření se bere rozdíl mezi datem začátku měření a aktuálním datem

Dny, které spadají do vypočteného intervalu, se v komponentě zobrazí jako tlačítka. Po vybrání jednoho z tlačítek dojde k načtení datových souborů pro vybraný datum. Následně je komponenta překreslena potřebnými daty.

### 4.5.7 Historie pacientů

Pro zpětné dohledání naměřených hodnot pacientů slouží sekce **Historie pacientů**. Tato část aplikace obsahuje přehlednou tabulku, která obsahuje seznam všech evidovaných pacientů v systému. Komponenta tabulky je realizována komponentou *DataTables* knihovny *jQuery*. Pacienty je možné řadit podle jména, data zahájení měření, data ukončení měření a počtu senzorů. Ke snadnému nalezení je dále možné využít vyhledávací pole tabulky.

Data pacientů zajišťuje kolekce *Biomon.patientCollection*. Použitím funkce *fetch*, je přes volání API získán seznam všech pacientů.

Po úspěšném získání dat jsou jednotlivé záznamy ve funkci *updateData* zformátovány do podoby, kterou akceptuje funkce *fnAddData*. Tato funkce slouží k přidání jednoho řádku do tabulky.

## 4.6 Implementace serverové části aplikace

Zdroj dat klientské části poskytuje serverová část aplikace. Čte a zapisuje data do databázových souborů. Jelikož aplikační logika je přesunuta do klientské části aplikace, serverová část pouze řídí přístup k datům. Z tohoto důvodu není tak rozsáhlá jako klientská část webové aplikace. Díky možnostem a efektivní práci s vybraným frameworkem je kód serverové části aplikace tak krátký, že jsem se rozhodl ho umístit do jediného souboru. V tabulce č. 1 je popis serverového API. Serverová část aplikace přistupuje ke třem SQLite databázím.

HTTP metoda	URI	Oprávnění	Popis
GET	/patient/active	Uživatel Administrátor	Vrací seznam záznamů pro aktuálně sledované pacienty.
GET	/patient/all	Uživatel Administrátor	Vrací seznam záznamů všech pacientů vedených v systému.
GET	/patient/:id	Uživatel Administrátor	Vrací záznam pacienta s číslem, které je zadáno jako parametr <i>id</i> .
POST	/patient	Administrátor	Vytváří záznam pacienta na základě parametrů odeslaných s požadavkem.
PUT	/patient/:id	Administrátor	Aktualizace záznamu pacienta s číslem, které je zadáno jako parametr <i>id</i> .
DELETE	/patient/:id	Administrátor	Odstraní záznam pacienta s číslem, které je zadáno jako parametr <i>id</i> .
GET	/sensor/selected	Administrátor	Vrací seznam senzorů, které byly označeny pro přiřazení.
GET	/sensor/info	Uživatel Administrátor	Vrací seznam informací o všech senzorech ( stav baterie, atd. ).

POST	/user	Administrátor	Vytváří nového uživatele na základě parametrů odeslaných s požadavkem.
GET	/user/active	Administrátor	Vrací seznam aktivních uživatelů.
GET	/user/:id	Uživatel Administrátor	Vrací informace o uživateli, jehož číslo je zadáno jako parametr <i>id</i> .
DELETE	/user/:id	Administrátor	Odstraní záznam uživatele s číslem, které je zadáno jako parametr <i>id</i> .
PUT	/user/:id	Uživatel Administrátor	Změna hesla uživatele s číslem, které je zadáno jako parametr <i>id</i> .

*Tabulka 1: Popis serverového API*

# 5 Testování a hodnocení dosažených výsledků práce

Tato kapitola popisuje test implementované webové aplikace z pohledu funkčnosti, vizuální stránky a kompatibility prohlížečů. V druhá část této kapitoly je věnovaná simulaci zátěže reálného systému.

## 5.1 Kompatibilita prohlížečů

Jedním z požadavků bylo ověření kompatibility webových prohlížečů. Kompatibilitu webových prohlížečů jsem ověřil v prostředí Windows 8.1 provozovaném na laptopu s rozlišením displeje 1440 x 900px. K ověření jsem použil následující prohlížeče:

- Mozilla Firefox verze 37
- Internet Explorer verze 11
- Google Chrome verze 42

Funkčnost webové aplikace jsem ověřil průchodem všemi jejími částmi. Po funkční stránce jsem nezjistil žádné rozdíly mezi testovanými prohlížeči. Aplikace je tedy kompatibilní se všemi testovanými moderními webovými prohlížeči.

Po vizuální stránce se aplikace v testovaných prohlížečích chová obdobně. Zjistil jsem sice drobné rozdíly ve vykreslení, ty ale neovlivnily funkčnost ani přehlednost aplikace. Rozdílné vykreslení je způsobeno použitím odlišných renderovacích enginů.

## 5.2 Simulace zátěže reálného systému

Abych otestoval reálné zatížení systému, nejprve jsem hledal vhodný nástroj pro tyto účely. Nástrojů pro otestování výkonnosti webových serverů je mnoho. Pro výběr jsem použil srovnání open source nástrojů [ 17 ]. Rozhodl jsem se použít nástroj Gatling [ 18 ] a to z toho důvodu, že ovládání tohoto nástroje je snadné a produkuje podrobné a dobře čitelné výstupy.

### 5.2.1 Gatling load testing framework

Je nástroj pro testování zatížení systému založený na jazyce Scala<sup>7</sup>. Gatling framework se skládá z 2 základních částí: *Gatling* a *Recorder*. Gatling zajišťuje provádění testování a také generuje přehledný výstup ve formátu HTML. Každý test, který nástroj Gatling spouští, je popsán pomocí skriptu. Tyto skripty jsou napsány v jazyce DSL (Domain-specific language<sup>8</sup>). Pro automatické vytvoření testovacího skriptu je možné použít nástroj Recorder. Po spuštění si Recorder ukládá veškerou aktivitu uživatele, která probíhá mezi webovým prohlížečem a serverem. Tímto způsobem je možné zaznamenat reálné chování uživatele.

### 5.2.2 Vytvoření testovacího skriptu

Gatling umožňuje simulaci více virtuálních uživatelů zároveň. Dále umožňuje souběžně simulovat různé chování uživatelů. Za účelem přesnější simulace a přiblížení se v co nejvyšší míře reálnému prostředí, jsem zaznamenal 3 varianty chování uživatele během používání webové aplikace:

- 1. varianta – uživatel se přihlásí do systému a následně se nachází na úvodní obrazovce (pouze sleduje vývoj měřených hodnot)
- 2. varianta – uživatel se přihlásí do systému a následně prochází historií

---

<sup>7</sup> <http://www.scala-lang.org/>

<sup>8</sup> [https://en.wikipedia.org/wiki/Domain-specific\\_language](https://en.wikipedia.org/wiki/Domain-specific_language)

měření u jednotlivých pacientů

- 3. varianta – uživatel se přihlásí do systému a následně přidává a odstraňuje pacienty

Další funkcí Gatlingu je možnost postupného spouštění simulovaných uživatelů. Tzn. pokud Gatling simuluje 10 uživatelů a doba postupného spuštění je 20 vteřin, tak se chování 3. uživatele začne simulovat po 6 vteřinách a 10. po 20 vteřinách. Kombinací různých vzorů chování uživatelů a postupného spouštění je možno docílit poměrně reálného používání systému.

### 5.2.3 Využití výkonu Raspberry Pi

Za účelem sledování výkonu Raspberry Pi během simulace jsem s využitím znalostí z předmětu KIV/ZOS a vytvořil Bash skript, který periodicky sleduje využívání výkonu Raspberry Pi. Vytvořený skript *monitor.sh* zapisuje každých 5 vteřin aktuální čas spolu s využití procesoru. Díky těmto informacím je možné zpětně dohledat, jaký dopad má prováděná simulace zátěže na výkon Raspberry Pi.

### 5.2.4 Provedení simulace

Simulace zátěže byla provedena na zařízení Raspberry Pi popsaném v kapitole 2.2 a laptopu Lenovo T520 s čtyř jádrovým procesorem Intel Core i7 2.70 GHz, 16 GB RAM, s pevným diskem SSD o velikosti 120 GB, s operačním systémem Microsoft Windows 8.1 Pro. Dále jen laptopu. Zařízení byla během simulace zátěže propojena ethernetovým kabelem délky 2 metry.

Pro simulaci zátěže byla provedena následující posloupnost kroků:

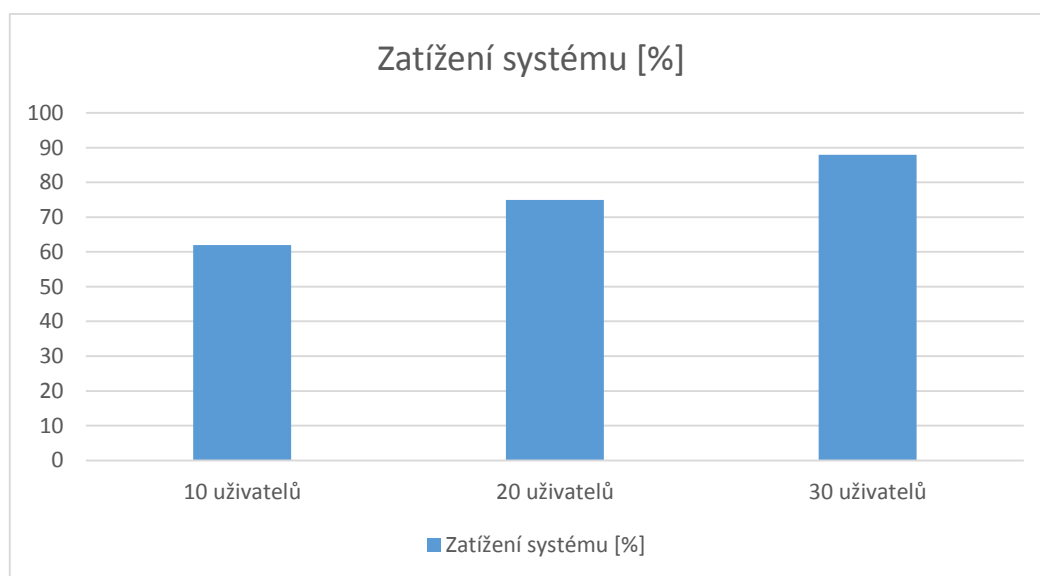
1. Na zařízení Raspberry Pi byl spuštěn monitorovací skript *monitor.sh*

2. Na laptopu byl spuštěn skript *RaspberrySimulation10.scala* použitím nástroje *Gatling* pro simulaci 10 uživatelů.
3. Po ukončení simulace na laptopu byl ukončen monitorovací skript na zařízení Raspberry Pi a naměřená data byla vyhodnocena.

Předchozí kroky byly opakovány pro simulaci 20 uživatelů (skript *RaspberrySimulation20.scala*) a 30 uživatelů (skript *RaspberrySimulation30.scala*). Každý z testů byl proveden pětkrát a naměřená data byla zprůměrována. Tabulka č. 2 obsahuje výsledky ze všech tří prováděných simulací. Naměřená data využití procesoru byla také zanesena do grafu č. 1 pro lepší orientaci.

Počet uživatelů	Průměrná odezva [ms]	Zatížení systému [%]
10 uživatelů	380	62
20 uživatelů	512	75
30 uživatelů	730	88

*Tabulka 2: Výsledky simulace*



*Graf 1: Zatížení systému*



### **5.2.5 Hodnocení výsledků simulací zatížení**

Z dosažených výsledků je patrné, že výkon zařízení Raspberry Pi je dostatečný pro provoz do 20 uživatelů při odezvě systému do 500 ms. Naměřená data však předpokládají neustálou činnost uživatelů. Z tohoto důvodu je pravděpodobné, že v reálném nasazení bude využití Raspberry Pi o něco nižší.

## **5.3 Omezení realizovaného řešení**

Jedním z bodů zadání bylo implementovat podpůrné funkce systému, umožňující zasílat varovné signály v případě definovaných situací, např. v případě, že dojde k překročení předem stanovených teplotních hodnot, prostřednictvím SMS, nebo e-mailem.

Zvolená architektura však neumožňuje bez aktivní činnosti uživatele monitorovat naměřené hodnoty za účelem identifikace situace, ve které by mělo nastat provedení předem definované akce (např. odeslání SMS nebo e-mailu). Z těchto důvodů není tato část zadání splněna.

Pro splnění i této části zadání by bylo potřeba vytvořit samostatnou aplikaci, která by průběžně monitorovala naměřená data bez součinnosti uživatele aplikace, a tím umožnila v reálném čase identifikovat situace, ve kterých má nastat provedení předem definované akce.

## 6 Závěr

Téma bakalářské práce jsem si zvolil zejména z důvodu, že mě zajímala možnost využití vybraných technologií v praxi. V průběhu jejího zpracování jsem si významným způsobem prohloubil znalosti webových technologií, především JavaScriptu, které jsem se do té doby sice používal, avšak nikoliv v takovém rozsahu, jaký vyžadovala tato práce. V rámci práce jsem se seznámil s novými technologiemi a zařízeními – Raspberry PI, nové knihovny: Backbone, DataTables, které považuji za velmi užitečné a určitě je použiji i v dalších projektech.

Po počátečních potížích, způsobených zejména omezenými zkušenostmi se specifiky asynchronního zpracování, se mi však podařilo vytvořit funkční aplikaci, která splňuje zadání.

Závěrem bych rád uvedl, že bakalářská práce je jednou z částí systému, který nabízí široké možnosti rozšíření, např. měření jiných fyziologických funkcí. Systém je nadále vyvíjen dalšími pracovníky NTC. Například došlo k vyvinutí jiné verze úvodní stránky ve formě dlaždic. Dále došlo i k rozšíření serverové části aplikace. Tyto nové funkce mohly být přidány bez nutnosti zásahu do současné architektury. Tuto skutečnost beru jako pozitivní signál o tom, že celé řešení bylo navrženo správně.

# Literatura, elektronické odkazy a zdroje

- [1] ROZMAN, Jiří. *Elektronické přístroje v lékařství*. Vyd. 1. Praha: Academia, 2006, 406 s., xxiv s. barev. obr. příl. Česká matice technická (Academia). ISBN 80-200-1308-3.
- [2] *Medical instrumentation: application and design*. 3rd ed. S.l.: John Wiley & Sons, 1998, xix, 691 s. ISBN 0471153680.
- [3] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi: uživatelská příručka*. 1. vyd. Brno: Computer Press, 2013, 232 s. ISBN 978-80-251-4116-8.
- [4] NORRIS, Donald. *Raspberry Pi: projekty*. 1. vyd. Brno: Computer Press, 2015, 264 s. ISBN 978-80-251-4346-9.
- [5] *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi* [online]. [cit. 2015-03-10]. Dostupné z: <https://www.raspberrypi.org>
- [6] Usage Statistics of Client-side Programming Languages for Websites, June 2015  
URL:  
<[http://w3techs.com/technologies/overview/client\\_side\\_language/all](http://w3techs.com/technologies/overview/client_side_language/all)>  
[ citováno: 2015-06-02 ]
- [7] Java Applet krok za krokem | Interval.cz  
URL: <<https://www.interval.cz/clanky/java-applet-krok-za-krokem/>>  
[ citováno: 2015-06-02 ]

[8] Adobe Flash

URL: < [https://en.wikipedia.org/wiki/Adobe\\_Flash](https://en.wikipedia.org/wiki/Adobe_Flash) >

[ citováno: 2015-06-02 ]

[9] Microsoft Silverlight – Wikipedie

URL: < [https://cs.wikipedia.org/wiki/Microsoft\\_Silverlight](https://cs.wikipedia.org/wiki/Microsoft_Silverlight) >

[ citováno: 2015-06-02 ]

[10] JavaScript Introduction

URL: < [http://www.w3schools.com/js/js\\_intro.asp](http://www.w3schools.com/js/js_intro.asp) >

[ citováno: 2015-06-02 ]

[11] Usage Statistics and Market Share of Server-side Programming Languages for Websites, June 2015

URL:

<[http://w3techs.com/technologies/overview/programming\\_language/all](http://w3techs.com/technologies/overview/programming_language/all)>

[12] ASP.NET – Wikipedie

URL: < <https://cs.wikipedia.org/wiki/ASP.NET> >

[ citováno: 2015-06-02 ]

[13] JavaServer Pages Technology

URL: < <http://www.oracle.com/technetwork/java/javaee/jsp/index.html> >

[ citováno: 2015-06-02 ]

[14] PHP: Hypertext Preprocessor

URL: < <http://php.net> >

[ citováno: 2015-06-02 ]

- [15] REST: architektura pro webové API – Zdroják  
URL:< <http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/> >  
[ citováno: 2015-06-02 ]
- [16] The Raspberry Pi Web Server Speed Test - Raspberry Pi Blog  
URL:  
< <https://www.jeremymorgan.com/blog/programming/raspberry-pi-web-server-comparison/> >  
[ citováno: 2015-06-02 ]
- [17] Open Source Load Testing Tools: Which One Should You Use? | BlazeMeter  
URL:  
< <https://blazemeter.com/blog/open-source-load-testing-tools-which-one-should-you-use> >  
[ citováno: 2015-06-02 ]
- [18] Gatling Project, Stress Tool  
URL: < <http://gatling.io> >  
[ citováno: 2015-06-02 ]

# Obsah přiloženého CD

Součástí této práce je CD, které obsahuje zdrojové kódy, knihovny a další nástroje pro test realizované aplikace. Struktura adresářů je následující:

- *readme.txt* – obsahuje popis adresářů a souborů na CD
- *www/* - obsahuje zdrojové soubory webové aplikace
- *doc/* - obsahuje text bakalářské práce (*BP\_JiriBrat.pdf*)
- *test/* - obsahuje skripty použité pro testování aplikace