VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Patrik Predný

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

HODOOR – ELEKTRONICKÝ DOCHÁZKOVÝ SYSTÉM
HODOOR – ELECTRONIC ATTENDANCE SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Patrik Predný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Edita Hejátková**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Patrik Predný ***ID:*** 154836
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Hodoor – elektronický docházkový systém

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte elektronický docházkový systém na míru pro použití ve společnosti ELEDUS s.r.o. Sestrojte klientský terminál pomocí hardwarové platformy Raspberry Pi. Tento terminál bude využívat RFID čtečku pro detekci osob, které budou k docházkovému systému přistupovat. Na platformě bude běžet aplikace pro komunikaci se serverem, který bude evidovat data o interakcích se systémem do databáze pomocí dosavadního serverového řešení. Pomocí webových technologií sestrojte aplikaci terminálu a front-end uživatelské rozhraní systému.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 25.5.2017

Vedoucí práce: Ing. Edita Hejátková
Konzultant:

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá problematikou evidence docházky. Cílem práce bylo navrhnout řešení a následně vytvořit hardwarový terminál elektronického docházkové systému. Docházkový systém je založen na nejnovějších webových technologiích s použitím hardwarového mini-počítače Raspberry Pi. Budoucí použití zařízení je možné jak pro domácí, tak pro komerční účely. Systém byl distribuován pod štítkem Open Source. Výsledkem práce je fyzický prototyp klientského terminálu založeného na platformě Raspberry Pi, aplikace terminálu založená na technologií Electron a webová aplikace komunikující se serverovým řešením.

KLÍČOVÁ SLOVA

Elektronický docházkový systém, evidence docházky, Raspberry Pi, Linux, Rasbian, webová aplikace, front-end, back-end řešení, HTML, CSS, JavaScript, Python, server-klient, Electron, Bootstrap framework, Open Source.

ABSTRACT

This thesis deals with the issue of attendance recording. The aim of the thesis was to design a solution and then create a hardware terminal for the electronic attendance system. The whole system is based on the latest web technologies using the Raspberry Pi hardware, for the client terminal. Future use of the device is possible for both home and commercial use. The whole system was distributed globaly as Open Source project. The result of thesis is a physical prototype of a client terminal, based on the Raspberry Pi platform, Electron based application for terminal and a web application communicating with the server solution.

KEYWORDS

Electronic attendance system, attendance registration, Raspberry Pi, Linux, Rasbian, web aplication, front-end, back-end solution, HTML, CSS, JavaScript, Python, server-client, Electron, Bootstrap framework, Open Source.

Bibliografická citace:

PREDNÝ, Patrik *Hodoor – elektronický docházkový systém*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky, 2017. 86 s. Diplomová práce. Vedoucí práce: Ing. Edita Hejátková.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Hodoor – elektronický docházkový systém“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 25.05.2017 .....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce paní Ing. Editě Hejátkové, za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování projektu. Dále bych chtěl poděkovat společnosti ELEDUS s.r.o., nejen za poskytnuté zázemí, vybavení a materiál, ale i za příjemné prostředí, pro vytvoření této práce. Poděkování patří hlavně kolegovi Ing. Ondřeji Vičarovi, za jeho ochotu a odbornou pomoc při řešení složitých úkolů spojených s touto prací. Také za projevenou důvěru pro realizaci jeho myšlenky, která se stala podnětem pro vznik této práce. V neposlední řadě však patří poděkování také mojí rodině, za podporu a pomoc při mojí tvorbě.

V Brně dne 25.05.2017 .....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešním světě plném výrazů jako jsou například Průmysl 4.0 (z ang. slova Industry 4.0), se setkáváme s pojmem IoT (Internet of Things - internet věcí). Tento pojem je také jedním z hlavních myšlenek výše zmiňovaného Průmyslu 4.0. Ve zkratce se jedná o propojení všech zařízení na společnou síť (nejlépe internet v globálním měřítku). Umožnit tak zařízením sdílet data mezi sebou nebo je odesílat do společné centrály. Další myšlenkou Průmyslu 4.0 je kompletní digitalizace. To znamená převádět „zastaralý“ analogový signál do digitální podoby pro zvýšení kvality dat.

Cílem této práce je vytvořit elektronický docházkový systém, který se vyznačuje právě výše uvedenými znaky. Celé zařízení má především usnadnit život lidem ve společnosti ELEDUS s.r.o. [1], kde do doby před zavedením elektronického systému fungovalo zastaralé „analogové“ zapisování příchodů zaměstnanců. Dalším ze splněných požadavků na zařízení, je výhradní komunikace pomocí bezdrátové sítě Internet. Klient postavený na hardwarové platformě Raspberry Pi bude pomocí internetového připojení schopen komunikovat se serverem. Na serveru bude běžet software obsluhující databázi s údaji o uživatelích.

Dá se říct, že toto zařízení splňuje požadavky kladené na industrializaci v dnešní době. Od počátku vývoje je kladen důraz na pozdější uvedení do světa pod štítkem Open Source (z ang. slova otevřený zdroj). Uživatelům z celého světa bude umožněn přístup ke zdrojovým údajům zmiňovaného zařízení, a budou tak moci přispívat k zlepšování celého vyvinutého systému.

Zadání práce se zaměřuje pouze na analýzu a následnou realizaci HW (Hardware) zařízení a dále vytvoření uživatelského tzv. „front end“[2] webového rozhraní systému. Nikoliv komplexním návrhem logiky, struktury databáze a „back end“ řešení serverové aplikace. Tyto prostředky budou v práci popsány jen okrajově.

[1] Zadavatel diplomové práce.

[2] front end/back end představují podstatné jména, front-end/back-end jména přídavné.

1 ZÁKLADNÍ ANALÝZA

Jedním z důvodů vzniku zadání této práce je digitalizace docházky zaměstnanců. Momentálně je ve společnosti ELEDUS s.r.o. [1] zhruba 30 zaměstnanců. Jedná se o 15 pracovníků na hlavní poměr, zbytek jsou studenti na brigádách a praxích. Evidence pohybu personálu je tedy důležitá nejen pro management, ale i v souvislosti s BOZP (Bezpečnost a ochrana zdraví při práci). Text níže, popisuje výňatek týkající se docházky a její evidence danou Zákoníkem práce.

„Podle ustanovení § 96 odst. 1 zákoníku práce je zaměstnavatel povinen vést u jednotlivých zaměstnanců evidenci s vyznačením začátku a konce odpracované směny, práce přesčas, další dohodnuté práce přesčas, noční práce a doby v době pracovní pohotovosti a dále evidenci s vyznačením začátku a konce pracovní pohotovosti, kterou zaměstnanec držel.“ [4]

1.1 Definice stávajících problémů

Na příkladě lze jednoduše ukázat, že evidence docházky je nezbytnou součástí každé firmy. Evidence docházky je běžnou praxí u velkých korporací, týká se samozřejmě i malých a začínající startup-ových firem.

V současné době se docházka ve společnosti řeší pomocí jednoduché metody tím, že každý z docházejících osob se zapíše do formuláře. Jedná se o papírovou formu evidence. Z počátku se toto řešení zdá býti dostačujícím, problémy však nastávají a mnohonásobně se zvětšují s nárůstem zaměstnanců.

Jedním z problémů je přepisování docházkové listiny do pracovního výkazu na konci měsíce. Tento proces je poněkud zdlouhavý. Dalším problémem je nedostatečná kontrola času obědových přestávek. Společnosti se setkávají i s pracovní cestou. Zaměstnanec se fyzicky nezdržuje v prostorách firmy. Ze strany BOZP to není problém, ale je nutné evidovat čas, který zaměstnanec věnuje v průběhu pracovní cesty k vykonávání práce.

1.2 Možnosti řešení problémů

Řešením této situace je buď koupit stávající systém, nebo si navrhnout vlastní. Při pohledu na finanční náročnost je stávající systém o něco dražší, ve výsledku však jednoduší. Návrh vlastního systému pro evidenci docházky je časově náročnější proces, přináší však možnosti vlastních úprav. Komerční řešení častokrát nepodporují

[1] Zadavatel diplomové práce

možnost úpravy dle požadavků zákazníka. Pokud tomu tak je, stává se tato varianta finančně náročná a ve výsledku se nevyplatí.

Firmy, které se přímo zabývají vývojem, nebo distribucí docházkových systémů nabízí i nezávazné ocenění uživatelských sestav. Budeme počítat zhruba 30 zaměstnanců. Každý z nich má k dispozici 1ks identifikačního klíče k přístupu. Centrální server s jedním klientem a napájecími zdroji. Výslední cena z nezávislého nacenění řešení od firmy GACC, spol. s r.o., vychází na 19.270 kč bez DPH.[5] Možnost systému na míru, jako i v tomto případě, je řešená jen volitelným množstvím SW licencí, terminálů (klientů), záložních zdrojů, přístupových klíčů...
Nejvhodnější a tedy i zvolenou variantou je samotný návrh zařízení. Tendence je vyvinout komplexní a modulární systém. Ten bude vytvořen na míru dle požadavků zadavatelské společnosti diplomové práce. Dále jeho rozšiřitelnost nebude závislá od třetích stran a tím nebude finančně náročná. Výňatek ze Zákoníku práce uvedený níže potvrzuje správnost tohoto rozhodnutí.

„Zákoník práce dále neupravuje, jakou formou má zaměstnavatel evidenci pracovní doby vést. Je tedy na uvážení zaměstnavatele, jak tuto povinnost na daném pracovišti naplní. Při kontrole ze strany inspekce práce je předložení evidence pracovní doby zpravidla jedním z prvních požadavků inspektora, neboť bez této není možné zkontrolovat správné odměňování zaměstnanců, dodržování limitů pracovní doby, práce přesčas a čerpání odpočinků v práci. Evidence pracovní doby musí být přehledná, průkazná a odpovídat účelů vedení evidence pracovní doby, jinými slovy inspektor, který kontrolu provádí a zaměstnanec, který má právo do ní nahlížet, musí být schopni z tohoto dokladu sestavit ucelený obraz o pracovní době jednotlivého zaměstnance.“ [4]

Forma evidence docházky zatím není zákonem jinak blíže specifikována. Jediný požadavek ze strany inspektorátu je, aby byla evidence přehledná, průkazná a odpovídala účelům vedení evidence pracovní doby.

1.3 Závěr ze zjištěných údajů

Vlastní návrh systému evidence docházky je možný a zákonem povolený. Je tedy zcela na zadavateli (tvůrcích), jakým stylem bude docházka vedena a jakou formou budou sestavovány měsíční výkazy práce jednotlivých zaměstnanců.

Ze zadání diplomové práce jasně vyplývá, že je vhodné použít hardwarovou platformu Raspberry Pi s dodatečným RFID modulem pro evidenci docházky. Na této platformě bude běžet klientská aplikace, založena na webových technologiích. Pomocí bezdrátové sítě WLAN bude komunikovat se serverem. Server bude dále uklá-

dat data do databáze a vhodným způsobem je zpracovávat. Hardwarová platforma Raspberry Pi a základy tvorby webových stránek jsou dále podrobněji rozebrány v této práci.

1.4 Požadavky pro systém

Společnost ELEDUS s.r.o. [2] si stanovila tyto požadavky na docházkový systém. Dle těchto požadavků se bude sestavovat logika docházkového systému Hodoor.
Jsou to převážně:

- Sestrojit fyzicky klientský terminál s aplikací, ke kterému budou uživatelé přistupovat pomocí RFID čipů nebo karet.
- Uživatelé mají na výběr základní akce, jako je záznam času příchodu a odchodu.
- Dále mají možnost zvolit si v průběhu pracovního dne neplacené volno v podobě pauzy na oběd, nebo hrazenou pracovní cestu.
- Z těchto událostí je vždy možné se vrátit zpět do normálního pracovního režimu.
- Pro administraci není důležitý přesný příchod zaměstnanců. Kontrolovat se bude jen standardní 8-hodinová pracovní doba.
- Uživatelé (zaměstnanci) mají dle smluv zavedenou flexibilní pracovní dobu.
- Systém bude umožňovat kontrolu v administrační sekci. Ta musí obsahovat seznam jmen a odpracovanou dobou zaměstnanců vztaženou na daný pracovní měsíc.
- Každý uživatel si musí do konce 5. dne následujícího měsíce vyplnit do systému projekt, kterému se danou pracovní dobu věnoval.
- K těmto účelům je vhodné pro uživatele sestrojit webovou aplikaci, která usnadní zadávání údajů do systému.
- Každý z uživatelů bude mít v systému uvedeno jméno, příjmení a bude mu přidělen RFID čip.
- Při tvorbě designových prvků používejte korporátní barvu firmy, nebo její odstíny. Firemní modrá: #00AEEF.

Jedná se tedy o „případový“ docházkový systém (z ang.: casual). Nejsou pevně stanoveny příchody a odchody zaměstnanců, které by se měly sledovat a dodržovat.

[2] Zadavatel diplomové práce

2 RASPBERRY PI

Raspberry Pi, zkráceně RPi, je označení série mini počítačů (velikosti kreditní karty, dle [1]). Aktivně jej vyvíjí společnost Raspberry Pi Foundation. Původně vznikla za účelem výuky programování Software a práci s Hardware na školách. [3] Postupem času získalo RPi dobrou pověst a je častou variantou řídící logiky v Open Source, DIY projektech, nebo projektech mladých firem. První generaci těchto mini počítačů reprezentoval model Raspberry Pi 1. Zhruba půl roku od vydání první verze RPi byl vyvinut operační systém Raspbian. Tento operační systém je dále popisován. Proto se s ním budeme později zabývat podrobněji.

2.1 Základní informace

Všechny modely Raspberry Pi jsou osazeny SoC Broadcom procesory s ARM jádrem (podobně jako dnešní mobilní telefony). [1] Dále obsahují integrovaný grafický procesor, operační pamět velikosti 256 MB až po 1 GB. Pro zavedení OS slouží slot pro SD karty. Většina má také k dispozici minimalně 1x USB slot, audio výstup, HDMI nebo kompozitní výstup videa. K základní komunikaci nebo k připojení dalších rozšíření slouží jednak Ethernetový konektor, nebo GPIO piny s podporovanou, například, I2C sběrnicí.[1] [3]

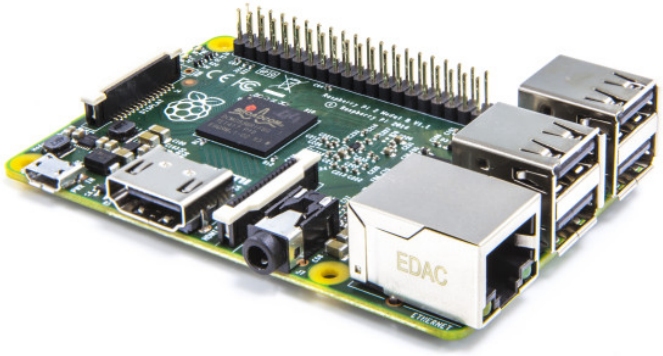
2.2 Raspberry Pi 3 Model B

Jedná se o nejnovější Hardware-ovou aktualizaci Raspberry Pi. Její dostupnost na trhu se datuje měsícem únor 2016. Dalším vhodným kandidátem je o rok starší verze Raspberry Pi 2 model B.
Třetí verze disponuje převážně:

- vyšší taktovací frekvencí (1.2GHz oproti 900MHz),
- procesor CPU s 64-bitovou architekturou,
- presenci integrovaného WLAN a Bluetooth modulu.

Nejvhodnějším kandidátem, nejen kvůli svému výpočetnímu výkonu, ale i pro presenci integrovaného WLAN modulu je tedy Raspberry Pi verze 3 (obrázek 2.1). Cena tohoto modelu se nijak výrazně neliší od předchozí verze RPi v. 2 model B.

[1] SoC - System on Chip



Obr. 2.1: Ukázka Raspberry Pi 3 Model B (převzato z [2]).

2.2.1 Klíčové parametry

Následující tabulka zobrazuje základní technické parametry Raspberry Pi 3 [18]. Jak bylo již uvedeno, tento model vychází z předchozí verze. Má tedy stejný form factor jako Raspberry Pi 2 model B. S předchozími verzemi je plně kompatibilní. [17]

Tab. 2.1: Základní parametry Raspberry Pi 3 model B [1] [17] [18]

Parametr	Raspberry Pi 3 Model B
Architektura	ARMv8 32/64bit
SoC	Broadcom BCM2837
CPU	ARM Cortex-A53, 4-jádrový
Rychlost CPU	1.2 GHz
Paměť	1GB SDRAM
GPU	Broadcom VideoCore IV @ 250 MHz
Video výstup	HDMI (rev 1.3), kompozitní 3,5mm TRRS
Sítové připojení	10/100 Mbps, 802.11n WLAN, BT 4.1
IO piny	17 GPIO
Počet USB	4
SD slot	MicroSDHC
Napájecí konektor	MicroUSB
Napájecí příkon	800 mA (4 W)

2.3 Operační systémy pro Raspberry Pi

Dle dostupných zdrojů momentálně existuje více než 7 různých OS, které lze spustit na platformě Raspberry Pi. Jsou to většinou komplexnější systémy založeny na Linux OS, nebo jednoúčelové systémy např. pro přehrávání médií. Výčet zajímavých OS pro RPi [19] [20]:

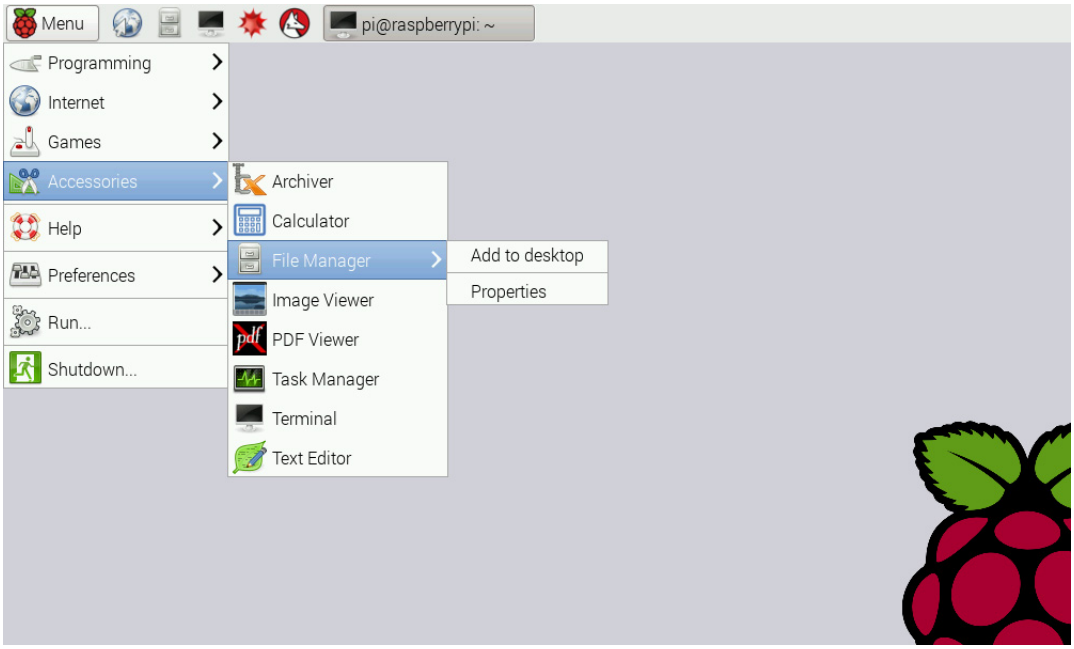
- *Raspbian* - **doporučuje se**, vychází z dist. Debian.
- *Pidora* - vychází z dist. Fedora.
- *Archlinux*.
- *OpenELEC* - dokáže spustit XBox Media Centre.
- *RaspBMC* - podobný OpenELEC.
- *Risc OS* - není postaven na Linuxu, RPi verze sestavená speciálně pro ARM procesory.
- *Firefox OS* - zatím ve vývoji, nepodporuje vstupní zařízení.
- *Android* - firma Broadcom pracuje na úpravě Android 4.0 pro použití v RPi.
- *Windows 10 IoT Core* - speciální verze známých Windows. Jedná se o vývojovou platformu pro IoT.

2.3.1 Raspbian Jessie

Základním OS pro Raspberry Pi je Raspbian. Tento operační systém není přímo vyvíjený společností Raspberry Pi Foundation. O aktivní vývoj se stará malá komunita vývojářů, převážně Mike Thompson a Peter Green. V podstatě se jedná o upravenou Linuxovou distribuci, vycházející z původního Debianu. [2]

Operační systém Raspbian je přímo určen pro použití v RPi, pro který je optimalizován. Tento operační sytém obsahuje sadu nástrojů pro obsluhu zařízení, pár programů a utilit, které lze spustit na RPi. Nejnovější distribuce Raspbianu má označení Jessie [3]. Na oficiálních stránkách Raspberry Pi, můžeme stáhnout hned dvě verze toho OS. Jsou to:

- *Raspbian Jessie with Pixel* - plná sestava, obsahuje i GUI (Graphical User Interface - grafické uživatelské rozhraní) Pixel.
- *Raspbian Jessie Lite* - zjednodušená (minimální) verze bez GUI, jen s CLI (Command Line Interface - příkazový řádek).



Obr. 2.2: Operační systém Raspbian Jessie s GUI Pixel.

Obsahuje mnoho balíčků a programů. Například programy pro edukaci, programování, ale i běžné používání [21].

[2] Debian - Linuxová distribuce
[3] Informace získány 11/2016

3 MODERNÍ WEBOVÉ TECHNOLOGIE

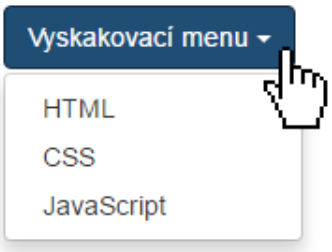
Obecně výraz framework představuje softwarovou strukturu, sloužící jako podpora při programování a vývoji aplikací. Hlavním cílem je převzetí typických problémů dané oblasti, čímž se usnadní vývoj. [32] Návrháři a vývojáři se můžou lépe fokusovat pouze na své zadání. Na podobném principu pracují i HTML a CSS frameworky.

Obsahují určitou strukturu HTML, pro jednoduché sázení a tvorbu aplikací. Dále také obsahují komponenty, které mají předdefinované vlastnosti. Programátor se nemusí tedy zabývat tvorbou struktury HTML, nebo přidáváním grafického vzhledu aplikace pomocí CSS. Nasázením těchto předdefinovaných komponent se práce kodéra podstatně urychluje a zjednodušuje. Tvůrce aplikace se tak může věnovat psaní programových funkcí.

3.1 Bootstrap framework

Jedná se o open source HTML, CSS a JavaScript framework. Momentálně [1] je dostupný pod MIT [33] licencí na službě Github [2]. Pomocí něj je tvůrce aplikace schopen sestavit jednoduché grafické rozhraní pro svou aplikaci.

Použití tohoto frameworku vyžaduje zkušenosti s HTML a psaním struktury webů. Do jednotlivých HTML tagů pomocí paramterů třídy (class) zadáváme stavebním blokům předdefinovaného CSS a JS funkce. Nabízí se také možnost použít už předem definované komponenty. Jako jsou například vysouvací seznamy s menu, bloky upozornění a pod ... Tyto komponenty mohou mít přiděleny JS funkce, například animace. Ukázka předefinovaného vyskakovacího menu ve frameworku Bootstrap je na obrázku 3.1.



Obr. 3.1: Komponenta Dropdown Menu z frameworku Bootstrap.

[1] Informace získány 11/2016

[2] Github - distribuované úložiště repozitářů. Pracuje s verzovacím systémem Git

Kód pro toto menu je následující:

```
<div class="container">
 <div class="dropdown">
  <button class="btn␣btn-primary␣dropdown-toggle"
        type="button" data-toggle="dropdown">
        Vyskakovací menu
  <span class="caret"></span></button>
  <ul class="dropdown-menu">
   <li><a href="#">HTML</a></li>
   <li><a href="#">CSS</a></li>
   <li><a href="#">JavaScript</a></li>
  </ul>
 </div>
</div>
```

Bootstrap je neustále vyvíjen (příchod verze 4), obsahuje mnoho možností si jej vlastnoručně poupravit. K tomu slouží *Sass* a *Less* preprocesory.

Výhod pro použití tohoto frameworku je hned několik. Podstatou je hlavně responzivní design výsledné aplikace. Tvůrce stránky vytváří rozměrově přizpůsobivé uživatelské rozhraní najednou, pro všechny typy zobrazovacích zařízení. Důvodů k použití právě Bootstrap frameworku je hned několik. Dalším důvodem je rychlé pochopení principů práce s tímto frameworkem. Jeho následná implementace do HTML kódu je velice jednoduchá. Obsahuje velké množství doplňků (předdefinované komponenty), se kterými se jednoduše a rychle pracuje. V neposlední řade je jeho velkou výhodou, možnost použití pro sestavení jak front-end, tak back-end UI prostředí/aplikace.

3.1.1 Sass a Less

Jsou CSS preprocesory [3]. Občas u tvoření CSS stylů nastává situace, kdy je potřeba definovat některé styly hromadně. Zařadit je do skupin - tříd, nebo aby některý ze stylů automaticky přebral vlastnosti jiného stylu - dědičnost. Tyto problémy řeší CSS preprocesory jako jsou například *Sass a Less*.

Prakticky zavádí programování do CSS. Sass je založen na programovacím jazyku Ruby, Less na JavaScriptu. Nevýhodou preprocesorů je nutnost kompilace, a proto se nedají zpracovávat v reálném čase. Existují ale různé nástroje, které umožňují integraci do textových editorů, nebo i serverů a umožňují automatickou kompilaci.

[3] Oficiální stránky http://lesscss.org/ a http://sass-lang.com/

Jejich využití je vhodné zejména pro ušetření času vývojáře. Po kompilaci, některým z volně dostupných kompilátorů [34] získáme CSS soubor s námi definovanými vlastnostmi.

Ukázka kódu napsaného v SCSS (Sass), konkrétně použití proměných:

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Po kompilaci získáme následující CSS kód:

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Často je možné měnit nastavení kompilátoru. To umožňuje kompilovat kód do tzv. „stlačeného“ (compressed) formátu, kdy je kód zapsán do souboru pouze v jediném řádku. Tento způsob generování je vhodný pro urychlení časů generace stránky prohlížeči.

3.2 Electron

Electron je multiplatformní (cross-platform) nástroj pro vytváření (programování) desktopových aplikací pomocí webových technologií (HTML, CSS a JS). V základu je umožněno programování pomocí NodeJS, nebo klasickém JavaScriptu. Je možné k němu doinstalovat různé softwarové balíčky (knihovny), pro podporu dalších funkcí.

Původně vznikl jako součást známého open-source editoru Atom. Dříve známý jako Atom Shell. Portál vývojářů Github vydal nakonec Elektron jako samostatný framework pro další uživatele. Na frameworku Elektron je postavených i mnoho dalších aplikací [36]. Jedná se například o:

- *Visual Studio Code* - IDE od Microsoftu.
- *Brave* - webový prohlížeč.
- *Slack* - kolaborativní platforma.
- *Gitkraken* - grafický nástroj pro zobrazení verzovacího jazyka git.
- *A další ...*

Framework Electron založený na Node.js, umožňuje překlad webových aplikací pomocí integrace do nástroje Chromium. Chromium je open-source webový prohlížeč, který položil základy známého prohlížeče Chrome od společnosti Google. Naproti Google Chrome je Chromium tzv. „osekanější“ verze. Chybí mnoho integrací, ale jeho podstatnou výhodou jsou mnohem menší nároky na HW [35].

Nejdůležitějším souborem v Electron-u je soubor *package.json*. Jedná se prakticky o popis dané aplikace. Obsahuje:

- Jméno aplikace (name).
- Verzi (version).
- Hlavní specifikace aplikace (main):
  - Rozměry a jiné vlastnosti okna (width, height, frame...).
  - Obsah menu chromia.
  - Mnohé další...
- Použité balíky (modules).

Existují také i jiné podobné způsoby jak psát desktopové aplikace pomocí webových technologií. Jsou to například frameworky NW.JS (Node-Webkit) nebo AppJS. Ty nesou ale řadu nevýhod, a nejsou delší dobu podporované tak velkou komunitou vývojářů [44].

Pomocí své API podporuje například i komunikaci s operačním systémem, konkrétně vytváření notifikací a oznámení pro uživatele mimo samotnou aplikaci. Nakonec je možné hotovou aplikaci zabalit, vytvořit tak instalátor pro OS Windows a jednoduše ji distribuovat [36].

Electron společně s webovými technologiemi a Raspberry Pi tvoří jednoduchý, ale účinný systém pro sestavení HW klienta pro docházkový systém. Nároky na spuštění aplikace napsané ve webových technologiích, spustitelných pomocí Electronu, nejsou velké. Proto i na platformě, jakou je i Raspberry Pi, se očekává plynulý běh aplikace a vysoká stabilita.

4 AUTOMATICKÁ IDENTIFIKACE A SBĚR DAT

Termín Automatická identifikace a sběr dat (z angl. slova Automatic Identification and Data Capture), neboli AIDC se používá pro identifikaci objektů a následný sběr dat na těchto objektech. Celosvětové rozšíření AIDC bylo umožněno zavedením norem pro informační technologie ISO/IEC JTC1 a IEEE 802. [8] U identifikace osob, jako v tomto případě, se nejčastěji setkáváme s dostupnými technologiemi identifikace. [6] Jsou to:

- Čárové kódy - bar codes (QR kód, EAN ...),
- RFID,
- NFC,
- Biometrie.

Tyto technologie se vyznačují především rychlou reakční dobou, přesnou a nenákladnou identifikací. [6] Některé, například biometrická identifikace, nevyžadují žádné identifikační prostředky k detekci navíc.

Technologie RFID a NFC vyžadují ke komunikaci pasivní (případně aktivní) prvek. V něm jsou uložena data pro sběr. Může se jednat přímo o ID (identifikační) tagy (štítky), nebo identifikační karty. Prakticky se jedná o totéž, jen v jiném tvarovém provedení. Setkáváme se i s označením transpondér. Seznam možných provedení RFID tagů:

- Paper Tag - RFID tag na papírové podložce, používá se k označování oblečení, výrobků ...
- EPC Tag - Electronic Product Code - unikátní struktura určená pro identifikaci položek v logistickém řetězci, není určen k přenášení osobních dat.
- Inlay Tag - určeny pro pozdější potisk, na papírovém, PET nebo PVC substrátu.
- Button Tag - v „knoflíkovém“ provedení.
- Metal Tag - v kovovém pouzdře, nejčastěji Lithium, pro HF a UHF provoz.
- Key Tag - nejčastější provedení ve formě klíčku.
- Glass Tube Tag - novinka, díky tomuto provedení se dají RFID tagy implementovat např. pod kůži.
- Ceramic Tag - na keramickém substrátu.
- Disc Tag - v diskovém provedení, tenčí než button tag.

4.1 Biometrické metody

Dalším způsobem jak identifikovat osoby, je využitím biometrie. Jedná se v podstatě o soubor metod pro verifikaci/identifikaci osob podle jejich fyziologických a morfologických znaků. Setkáváme se i s biometrickými systémy založenými na indikaci osob podle jejich behaviorálních znaků [9].

4.1.1 Biometrické znaky

Mezi základní biometrické znaky, používané v praxi patří:

- otisk prstu,
- obraz tváře,
- rozměr ruky,
- vlastnosti duhovky,
- nebo sítnice oka.

Snímání a vyhodnocování behaviorálních vlastností osob je podstatně náročnější, a pro rychlou identifikaci osob nemá prakticky význam [9] [10]. Setkáváme se s kombinací vícero biometrických systémů.

Na rozdíl od technologie NFC a RFID, biometrie nevyžaduje žádné dodatečné identifikační klíče (tagy nebo karty). Využívá přímo unikátní parametry lidského těla [10].

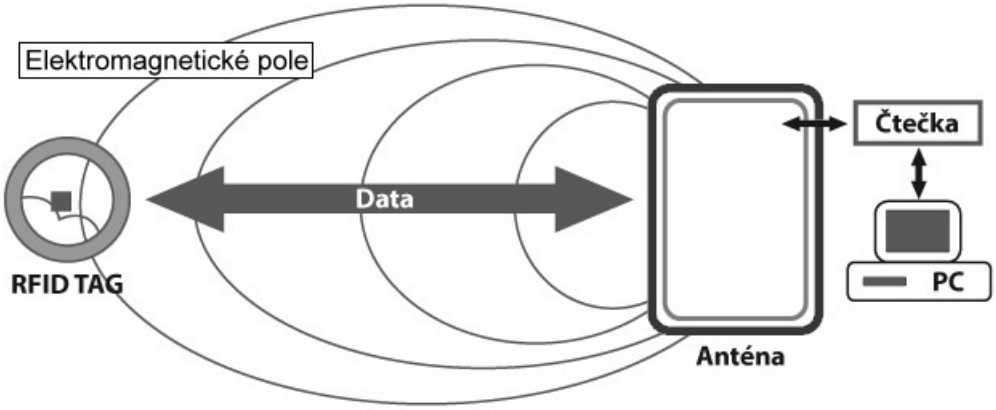
4.2 NFC identifikace

NFC - Near Field Communication je způsob bezdrátové komunikace na krátkou vzdálenost mezi elektronickými systémy (Short-Range Wireless Technology) [8]. Komunikace probíha mezi aktivním vysílačem a pasivním prvkem.

Vysílač komunikuje na frekvenci 13.56 MHz. Vzniklé EM pole (Electro Magnetic - elektro magnetické) napájí pasivní přijímač. NFC štítek (také označován NFC tag, případně karta) nepotřebuje žádný napájecí zdroj, energii získává přímo ze vzniklého EM pole. [7] Do samotných NFC tagů je možné tímto způsobem nahrávat jakékoliv data a následně je získávat, mazat nebo upravovat.

4.3 Úvod do problematiky RFID

Technologie RFID (Radio Frequency Indentification) je jednou z AIDC metod. Navazuje na technologii čárových kódů (2D struktury - čárové kódy, QR kódy ...), která včetně hardware společného s RFID přežívá v obchodních řetězcích.



Obr. 4.1: Zjednodušený princip komunikace RFID.

Výhodou, oproti čtení čárových (2D) struktur, je současná komunikace čtecího zařízení s několika RFID tagy najednou. Jedná se o hromadné čtení. Kdy je možné načíst informace z několik stovek RFID tagů za minutu. [12] U 2D struktur a jejich identifikace se musí získávat informace z každého čárového kódu samostatně.

4.3.1 Princip komunikace

Princip komunikace je blízký technologii NFC. Informace o sledovaném objektu jsou zaznamenány do RFID tagu (identifikační klíč, nosič). Ten obsahuje komunikační čip s pamětí a anténou. Čtecí zařízení (čtečka RFID) při komunikaci přenáší energii pomocí EM vln pro napájení RFID tagu. Při vzájemné komunikaci se využívá princip zátěžové modulace. Tag jako přijímač, odebírá určité množství energie z elektromagnetického pole čtecího zařízení a následně ho využívá pro zpětnou komunikaci.

Největší uplatnění nachází technologie RFID v logistice, výrobě, sledování objektů - logistických jednotek (zboží, palet, kontejnerů), sledování majetku, sledování zavazadel na letištích a evidence osob. [11]

4.3.2 Rozdíl mezi RFID a NFC

Jak bylo již uvedeno výše, technologie NFC vychází z RFID. Analogicky jsou si tedy velice podobné, avšak existuje pár rozdílů.

RFID tagy můžou být aktivní nebo pasivní. Aktivní RFID tag obsahuje vlastní zdroj energie a může vysílat i přijímat data na vzdálenost 100m. Tento princip je ideální pro použití v průmyslu a logistice. Pasivní nemají vlastní zdroj a energii získávají z vyzařovacího výkonu čtečky. Pracují na frekvencích:

- LF (Low Frequency) - 125-134 kHz,
- HF (High Frequency) - 13,56 MHz,
- UHF (Ultra High Frequency) - od 856 MHz do 960 MHz.

Zatímco NFC technologie komunikuje pouze na frekvenci HF, tedy 13,56 MHz. NFC, považována za bezpečnější, vyžaduje malou vzdálenost (několik jednotek cm) mezi komunikujícími zařízeními. Režimy NFC:

- peer-to-peer (P2P),
- card emulation (emulace karty),
- reader/writer (čtení a zápis).

RFID technologie využívá pouze režim reader/writer, pro čtení nebo případně zápis informace do paměti RFID tagu. V režimu Peer-to-peer (P2P) je možné být buď NFC tagem nebo NFC čtečkou. U RFID toto není možné. Využití mají převážně v mobilních zařízeních. Zde si pomocí NFC komunikace vzájemně sdělují informace, například kontakty, poznámky, data .... [7]

4.3.3 Výběr RFID tagů a čtečky

RFID je zatím nejpoužívanější technologií v evidenci produktů a osob. Proto je její použití pro docházkový systém vhodné. Výhodou je také rychlost a přesnost.

Na trhu je možné dostat čtečky v podobě samotných čipů, nebo jako celé zařízení. Pro účely sestrojení klienta založeného na Raspberry Pi s čtečkou RFID, bude nejlepší variantou použití hotového řešení. Jedním z nich je použití RFID čtečky s výstupem na USB s funkcí emulace klávesnice. Ta bude posílat hodnotu načtenou z RFID tagu přes USB přímo do klientské aplikace ve formě posloupnosti ASCI znaků. Tato možnost má především výhodu v její jednoduchosti. Není potřeba složitá instalace driverů (ovladačů) a nastavování pro fungování v RPi.

Pro identifikaci zvolíme pasivní RFID tagy. Není nutné, aby měly vlastní zdroj energie. RFID tagy se prodávají jako příslušenství k RFID čtečkám, nebo je lze dokoupit samostatně. Výhodou je provedení tagu jako klíčenka s přívěskem. Karty

se mohou často mechanicky poškodit. Samotný RFID tag je umístěn v ochranném obalu z plastu, obrázek 4.2.

Obr. 4.2: Jedno z možných provedení zapouzdření RFID tagů.

RFID čtečka ACM08M(N) komunikuje na frekvenci *f=125* kHz s výstupem USB, pracující v módu emulace klávesnice (obrázek 4.3). Výhodou je dobrá dostupnost a cena. V e-shope u distributora součástek SOS electronic je možné zakoupit tuto čtečku v hodnotě 874,02 Kč [15].

RFID tag musí být schopný pracovat se stejnou frekvencí. Vhodná je např. klíčenka RFID s označením ACM-ABS003BU, s cenou 19,70 Kč. Tato kombinace je nízko-nákladovým řešením a ideálním kandidátem pro sestrojení prototypové verze HW klienta docházkového systému.

Obr. 4.3: Ukázka hotového řešení RFID čtečky (převzato z [15]).

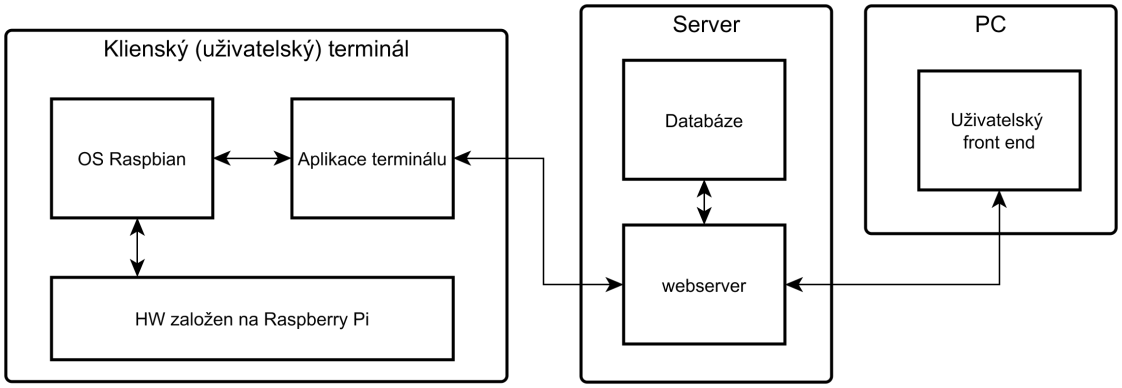
5 NÁVRH ŘEŠENÍ

V první fázi je potřebné navrhnout HW sestavu se všemi periferiemi určenou pro testovaní příslušného softwaru. Jedná se o uživatelský terminál, na kterém bude spouštěna aplikace pro odesílání dat na server. Dále je potřebné sestavit návrh uživatelského rozhraní pro sledování záznamů uživatelů.

5.1 Návrh blokových schémat

Tato kapitola popisuje návrh blokových schémat. Schéma celého systému, dále schéma pro sestavení klientského terminálu z HW hlediska a napájecích částí.

5.1.1 Blokové schéma systému

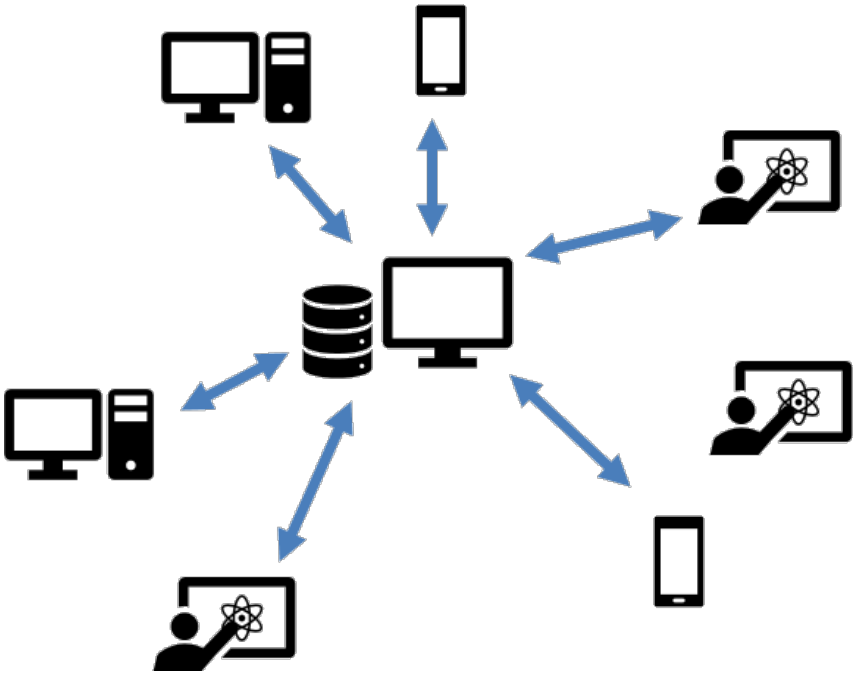


Obr. 5.1: Principiální blokové schéma systému.

Principiální blokové schéma systému z obrázku 5.1, popisuje komunikační bázi. Základem je server, komunikující s klientským terminálem. Server je webový server (webserver) na kterém běží tzv. WSGI služba (Web Server Gateway Interface) zabezpečující implementaci a použití Pythonu, programovacího jazyka, jako komunikačního a aplikačního nástroje pro webserver.

Terminál posílá na server údaje o uživatelských interakcích („jednotlivá pípnutí“). Údaje se ukládají do databáze. Dále se zobrazují na uživatelském počítači, v podobě webové stránky (front-end uživatelské rozhraní).

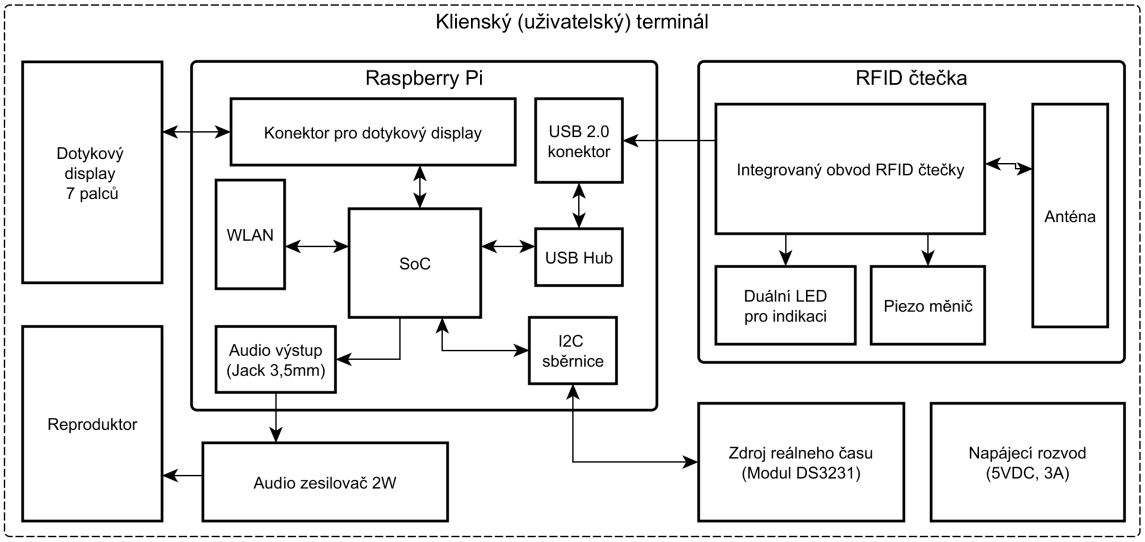
Komunikaci zabezpečuje využití technologie REST API, která pracuje s HTTP dotazy (HTTP Requests). Zpracovává dotazy typů GET a POST. Pomocí této API lze připojovat vícero klientů a uživatelských rozhraní k samotnému serveru. Obrázek 5.2 popisuje principiální systém komunikace. Zařízení s webovým uživatelským rozhraním (ikona počítačů, notebooků a smart-telefonů), a aplikace klientských terminálů (panáčci ukazující na electron aplikaci) komunikují se serverem s databází (počítač s ikonou databáze).



Obr. 5.2: Principiální zobrazení možné komunikace.

5.1.2 Blokové schéma HW terminálu

Obrázek 5.3 popisuje blokové schéma elektrického zapojení klientského terminálu. Terminál založen na Raspberry Pi verze 3, je vhodným řešením díky integrované WLAN komunikaci. Není tedy nutné připojovat externí bezdrátovou síť, například v podobě USB Wifi Dongle.



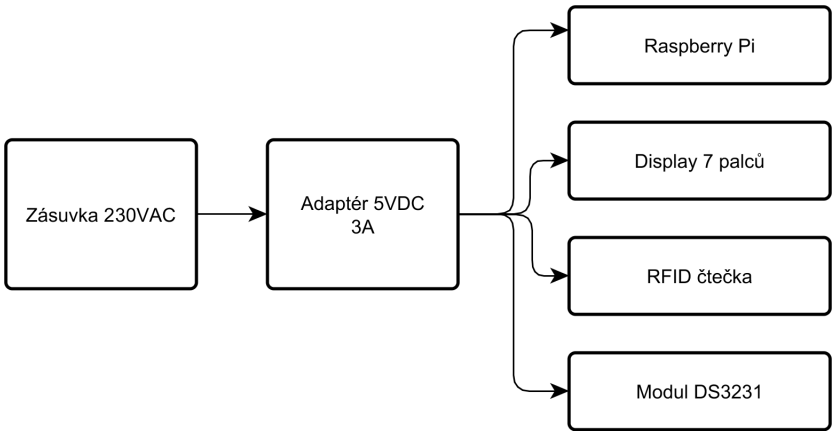
Obr. 5.3: Blokové schéma HW klientského terminálu.

Zobrazovací jednotkou je 7 palcový dotykový display. Ten byl přímo vyvinut pro připojení k Raspberry Pi v. 2 až 3 [37]. Výhodou je nepotřebnost externích ovládacích prvků, jako jsou například počítačové myši nebo klávesnice. Dotykový display je vhodným moderným řešením právě pro takto koncipovaný projekt.

Z audio výstupu realizovaného pomocí konektoru Jack 3,5mm je signál vyveden na audio zesilovač s výkonem 2W. Dále je k zesilovači připojen reproduktor. Tento zesilovací prvek bude určen pro případné přehráváni zvuků terminálové aplikace.

RFID čtečka, s výstupem pomocí USB s funkcí emulace klávesnice, obsahuje piezo měnič a duální LED pro indikaci stavu [15]. Indikační LED svítí červeně v idle stavu. Stav kdy se neděje žádná interakce. Při přiložení RFID Tagu LED blikne zeleně a piezo měnič vydá krátké pípnutí. Čtečka pracuje na frekvenci $f = 125kHz$, dokáže zaznamenávat přiložený RFID Tag do vzdálenosti přibližně 5cm od antény.

Posledním blokem je zdroj reálného času realizován modulem DS3231, komunikující s Raspberry Pi prostřednictvím I2C (TWI) sběrnice. Raspberri Pi obsahuje přímo knihovny pro použití tohoto modulu reálného času. Tento prvek je prozatím nevyužit podobně jako audio zesilovač. V další fázi projektu bude funkce reálného času důležitá zejména pro offline mód terminálu. Je známo, že platforma Raspberry Pi získává informaci o čase přes síťové rozhraní.



Obr. 5.4: Blokové schéma rozdělení napájecích větví.

Ze specifikací všech HW komponent vyplývá nutnost použití externího napájecího zdroje s výstupním napětím 5V DC. V případě použití dotykového 7 palcového displaye, musí být napájecí zdroj schopen pokrýt zhruba 2A odběr displaye a RPi. Použitím 5V zdroje s výstupním proudem 3A, získáme dostatečné napájení.

Napájecí strukturu popisuje obrázek 5.4. Vyhovující distribuce napájecího napětí zaručuje hladký běh systému. RPi má pro napájení určen microUSB konektor, podobně je tomu tak i u displaye. Display je možné napájet přes RPi GPIO. Příhodnější způsob napájení displaye je rozdělení napájení, a napájet display samostatně pomocí microUSB konektoru.

5.2 Návrh uživatelských aplikací

V případě front-end řešení UI se jedná o webovou stránku založenou na výše uvedených technologiích. Základem je samozřejmě klasické HTML, CSS a JavaScript. Prakticky se jedná o uživatelské prostředí, kde si mohou jednotliví uživatelé prohlížet svoje záznamy.

5.2.1 Popis struktury záznamů

Pro obecné chápání aplikací je potřebné si ujasnit některé termíny. Společně s vedoucím této práce panem Ing. Ondřejem Vičarem [1], vznikla následující struktura konceptu záznamů z terminálu. Každé pípnutí (interakci záznamu) budeme nazývat *swipe*.

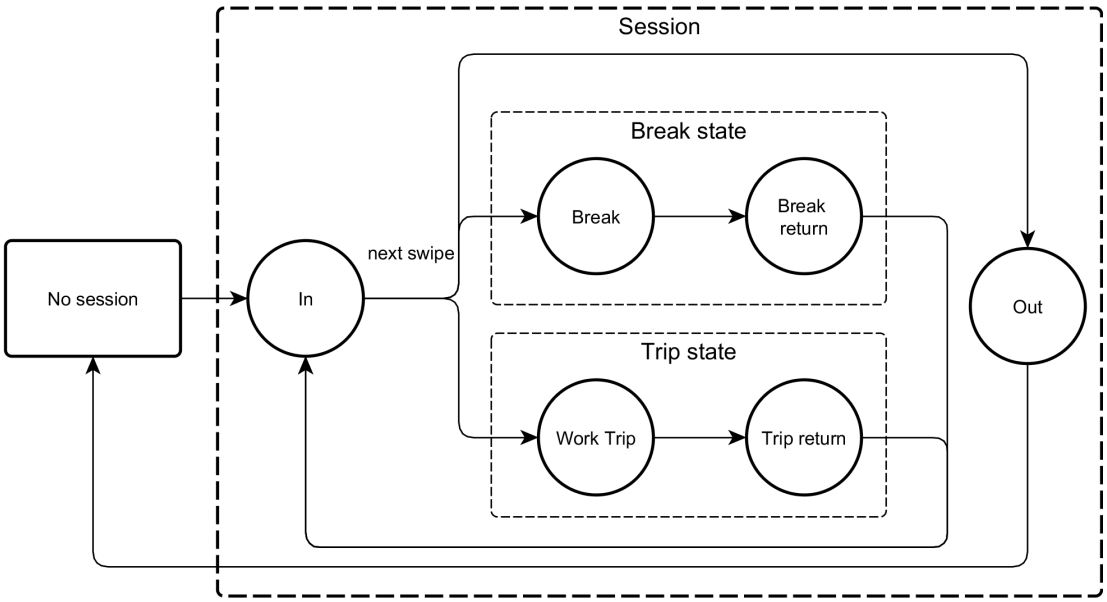
[1]Spolumajitel společnosti ELEDUS s.r.o., která je zadavatelem Diplomové práce

Rozeznáváme tyto druhy swipe-ů:

- *Incoming* - zkráceně In swipe, značí příchod do práce.
- *Outgoing* - zkráceně Out swipe, značí odchod z práce.
- *Break* - značí přestávku.
- *Break return* - návrat z přestávky.
- *Work trip* - zkráceně Trip, značí pracovní cestu.
- *Trip return* - návrat z pracovní cesty.

Rozsah mezi In a Out swipem, nazývejme *session*. Vycházíme z běžného života a požadavků na funkcionalitu zadavatelské společnosti. Následující příklad popisuje běžné používání (graficky obrázek 5.5).

Uživatel bez započaté session, může zvolit na terminálu pouze swipe typu In. Značí příchod do práce. Po další interakci na terminálu (další pípnutí) má na výběr swipy: Out, Break a Trip. Může si tedy zvolit jestli session ukončí (odchod z práce), nebo si zvolí mezi pracovní cestou a neplacenou přestávkou. Po odpracování několika hodin, nastane čas oběda. Uživatel si na terminálu vybere swipe s označením Break.



Obr. 5.5: Grafické znázornění swipe cyklů a povolovací logiky.

Po další interakci je možné zvolit jediný swipe a to Break return. Čas mezi Break a Break return se nezapočítáva do času stráveného prací. V případě aktivace Trip, je další povolený swipe Trip return. Podobně jako u Break stavu. Celý Trip stav se ale liší od Break stavu tím, že se započítáva do pracovního času. Může se jednat

například o pracovní cestu, obchodní cestu a jiné .... Pro ukončení session, uživatel zadá Out swipe. *Pozn.: Jeden pracovní den se může skládat z vícero sessions.*

5.2.2 Uživatelské skupiny a práva

Protože všichni uživatelé nemohou být ve stejném postavení. Je nutné rozdělit je do skupin. Nejjednodušším způsobem je rozdělení na:

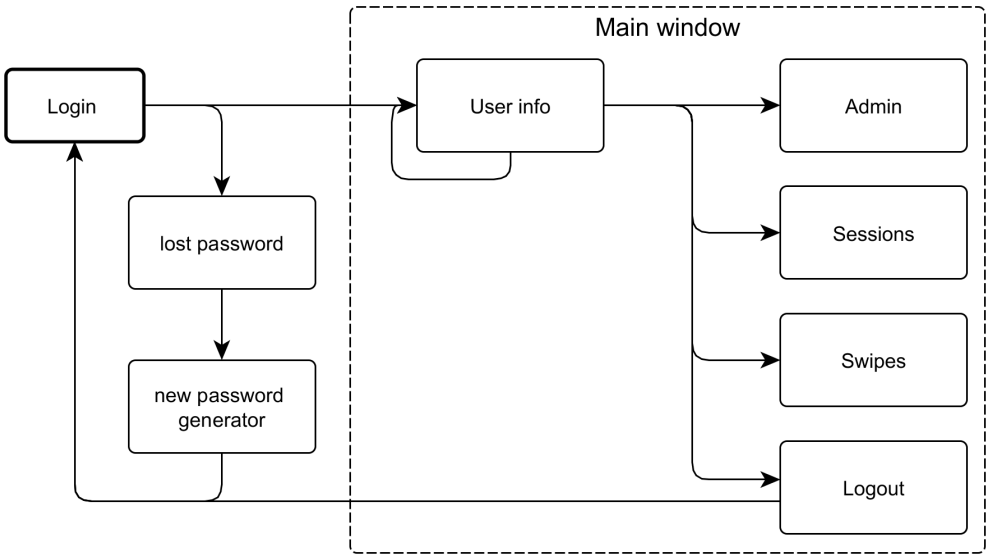
- *Admins* - administrátoři.
- *Users* - běžní uživatelé.

Běžný uživatel nemá právo prohlížet si záznamy jiných uživatelů, má možnost vidět jen ty svoje. Podobně nemá přístupové právo do back-end rozhraní (neboli úpravu záznamů v databázi). Naproti tomu administrátoři mají veškerá práva. Například přidání nového uživatele, úpravu jeho záznamů a záznamů jiných uživatelů.

Administrátoři mají za úkol dohlížet na záznamy jiných uživatelů, kontrolovat měsíční výkazy a zabezpečovat veškeré náležitosti s úpravami dosavadních nebo nových uživatelů.

5.2.3 Návrh front end řešení

Předtím než započne vývoj GUI webové aplikace, je dobrým zvykem si definovat strukturu stránek a podstránek. Jednoduchý koncept struktury je vyobrazen na obrázku 5.6.



Obr. 5.6: Návrh stránek a podstránek front end rozhraní.

Vysvětlení konceptu struktury je následující. V základním stavu uživatel není přihlášen. Proto po zadání url adresy do prohlížeče, kde se nachází tato webová aplikace, se dostane do přihlašovací (login) stránky. Zde má možnost zadat přihlašovací údaje, v opačné případě požádat o změnu hesla.

Je-li na stránce se změnou hesla, po zadání emailu obdrží odkaz se stránkou, kde si nastaví heslo nové. Zadá-li do přihlašovacího formuláře správné jméno a heslo, dostane se do stránky, kde jsou zaznamenány základní informace o uživateli (user info). V této stránce může setrvat, nebo se dostat do dalších.

Admin stránka je viditelná pouze pro administrátory. Zde bude měsíční souhrn hodin přiřazených ke každému uživateli. Stránka Session, obsahuje swipes z kterých se skládá daná session. Podobně stránka s názvem Swipes obsahuje seznam všech swipe-ů provedených uživatelem. Po kliknutí na Logout stránku se uživatel odhlásí ze systému.

6 PRÁCE S VERZOVACÍM SYSTÉMEM GIT

Tato krátká kapitola popisuje práci s verzovacím systémem Git a on-line bází repositářů Github.com. Každý program, projekt nebo jen jeho samotná část, by se měla zálohovat, nebo-li verzovat.

6.1 Jazyk Git

Git je distribuovaný systém správy verzí těchto výše zmíněných objektů, kterého tvůrcem je Linus Torvalds. Git je zdarma a je open-source. Je jednoduchý pro naučení a hojně používaný nejen v open-source ale i komerčním světě. Funguje na principu *snapshots*, dá se definovat jako snímkování daného souborového systému, nikoli jako jiné VCS systémy, které uchovávají pouze změny v souborech (verzích).

6.1.1 Základní pojmy

Pracujeme lokálně, tj. provádíme změny na lokálních souborech (*working directory*) a pro úspěšném ukončení našich změn (jako např.: přidání nové funkce do programu a pod.), můžeme přistoupit do tzv. *staging area*. V této oblasti jsou uloženy soubory, do kterých jsme zasahovali (soubory zdrojových kódů a pod.). Poté po příkazu *commit* nebo-li potvrzení změn se data můžou přenést do repositáře. Repositář je ve zjednodušeném významu skladiště našeho lokálního projektu, do kterého zasahujeme. Po ukončení práce v lokálním repositáři je možné data přenést do vzdáleného repositáře[38]. Vzdálený repositář může být umístěn na serveru Github.com.

6.1.2 Zprovoznění pod OS Windows

Pro další možnost práce s Gitem, je nutné ho mít nainstalován na našem počítači. Instalační balíček pro Windows, lze jednoduše získat jeho stažením z internetu. Při instalaci je nutné zvolit možnost: *přidat do systémové cesty (add git command to system path)*. Tímto je možné pracovat s gitem pomocí příkazové řádky, tj. bez grafického prostředí.

6.1.3 Základní příkazy

Tato podkapitola je určena pro seznámení se se základními přikazy git, tak aby bylo možné pokračovat na práci docházkového systému.

Nastavení uživatele v našem gitu:

```
git config --global user.name "patrik.predny"
git config --global user.email patrik.predny@github.com
```

Založení lokálního repositáře ve složce počítače (*working directory*):

```
git init
```

Přidání souborů do staging area:

```
git add <soubor>
git add --all //přidá všechny soubory
```

Uložení změny (*commit*):

```
git commit -m "text␣změny"
```

Vložení na vzdálený repositář:

```
git push origin <název větve kam chceme uložit>
```

Klonování existujícího projektu:

```
git clone <adresa vzdáleného repositáře /
 cesta v lokálním systému>
```

Zjištění stavu a posledních změn:

```
git status //vrátí výpis modifikovaných souborů
git log //vrátí výpis všech změn
git log -<číslo> //vrátí výpis posledních n změn
```

Toto jsou základní příkazy pro práci s gitem v projektu. Další příkazy jsou uvedeny v literatuře [39]. Je zde uvedeno jak lze pracovat s různými verzovacími větvemi.

6.1.4 Projekt Hodoor na Github.com

Jelikož projekt *Hodoor - docházkový systém*, je plánován jako open-source projekt, je důležité aby byl přístupný globálně. Pro tento účel je nejlepší použít některou ze služeb, která poskytuje uložení našeho vzdáleného repositáře. Nejznámějším je Github.com, kde je mnoho projektů, od různých uživatelů. Lze zde nastavit jak privátní, tak veřejný repositář.

Tento krok zabezpečí nejen transparentnost projektu, ale dá i jiným uživatelům možnost k němu přistupovat, přispívat do zdrojového kódu nebo jen přispívat svými komentáři.

Veřejný repositář umístěn na adrese Github.com nalezneme pod názvem projektu: **Hodooor** [40]. Tato projektová složka slučuje všechny repositáře týkající se Hodoor-u do jednoho projektu.

7 WEBOVÉ UŽIVATELSKÉ ROZHRANÍ

Tato kapitola popisuje webové uživatelské rozhraní, nebo-li front end rozhraní. Které slouží pro zobrazení údajů z databáze serveru pro potřeby uživatelů.

7.1 Úvod do problematiky

Webové uživatelské rozhraní je založeno na čistých webových technologiích (tzn. bez použití prostředí Electron). Jsou nimi HTML5,CSS3 (SCSS) a JavaScript s knihovnou jQuery. Systém zobrazuje informace pomocí back-end rozhraní.

To pracuje s databází serveru, komunikuje s klientským terminálem a zpracovává data pro front end. Back end byl napsán Ing. Ondřejem Vičarem [1] v jazyce Python za použití frameworku Django. Django je python framework pro práci s webovými servery, databázemi a pod. Využívá databázi SQL-Lite.

Back end je v této práci popsán jen okrajově, jsou objasněny základní pojmy v kapitole 5 odstavec 5.2.1.

7.2 Django Back end

Django, podobně jako jiné frameworky např. Nette pro práci s jazykem PHP, využívá pro vykreslování (rendering, renderování) stránek šablony. Šablony jsou určeny pro dynamické vykreslení stránek. Do šablon se vkládají funkce, proměnné a dynamicky generované bloky pomocí Django šablon (Django Templates).

Upravená HTML šablona pro vygenerování obsahu v Django frameworku může vypadat následovně:

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
</ul>
```

Tímto stylem jsou vytvořeny i šablony pro front-end rozhraní v Hodoor-u. To znamená, že každá HTML šablona obsahuje mimo klasických HTML značek také například dynamické proměnné, kterých obsah je generován a vykreslen ve front end-u na základě dat z back-end rozhraní. Více informací obsahuje odstavec 7.3.1.

[1] Konzultant diplomové práce a spolu-zakladatel projektu Hodoor

7.2.1 Popis a struktura souborů

Struktura souborů vytvořeného back-end rozhraní. Složka **static** obsahuje statické soubory, které se vyrenderují na počátku a uloží se na **root** složku serveru. Obsahuje převážně CSS styly, obrázky a JS funkce.

Údaje v nich obsažené jsou statické a v průběhu generování obsahu stránek se nemění. Výhodou je rychlý přístup k těmto souborům, to zrychluje i generovací čas stránek a podstránek.

Složka **templates** obsahuje právě zmiňované soubory šablon. Blíže jsou popsány v odstavci 7.3.1.

```
attendance
├── migrations
├── static ..................................... Statické soubory front end-u
├── templates .................................. Šablony front end-u
├── management
│   └── commands ............................... Obsahuje soubory s příkazy
│       ├── fill_mails.py
│       └── send_invites.py
├── admin.py ................................... Administrátorské funkce
├── apps.py
├── factories.py
├── forms.py ................................... Formuláře a jejich funkce
├── managers.py
├── serializers.py
├── tests.py ................................... Testování a funkce testů
├── utils.py
└── views.py ................................... Funkce pro vykreslování
```

7.2.2 Administrátorské rozhraní

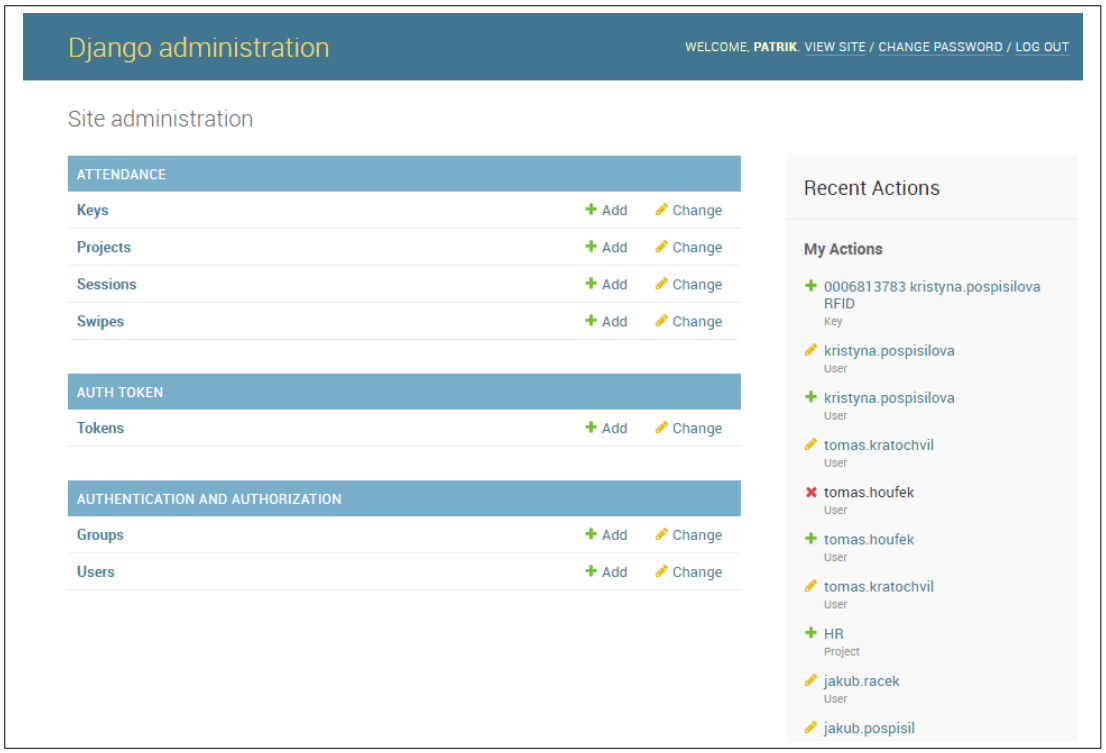
Jelikož se pracuje s databází je vhodné mít k dispozici prostředí vhodné pro běžné uživatele, kteří budou k systému přistupovat jako administrátoři. Ti mají za úkol spravovat a udržovat systém z hlediska uživatelských dat. Systém je nastaven proto tak, aby bylo možné chyby způsobené uživateli jednoduše odstraňovat bez nutnosti složitého programování apod. Pro tento účel slouží grafické prostředí pro správu databáze, které je automaticky generováno back-end rozhraním. Je použita hlavní (tzv. core) funkce frameworku Django.

Prostředí umožňuje:

- **Vytvářet uživatele a skupiny uživatelů,**
  - nastavit uživatelské jméno,
  - heslo,
  - kontaktní email.

- **Spravovat a měnit jejich údaje,**
  - jméno a příjmení,
  - přidat/odebrat uživatelům administrátorské oprávnění,
  - smazat uživatele.
- **Práce s projekty,**
  - vytvářet, mazat a měnit projekty.
- **Práci se Swipe-y a Sessions,**
  - upravovat a mazat jejich data.
- **Práci s přístupovými klíči,**
  - přidávat nové RFID klíče,
  - přiřazovat klíče uživatelům,
  - odebírat dosavadní klíče uživatelům, nebo je celé smazat (v případě, že je klíč hardwarově poškozený).
- **Získat token pro klientský terminál.**

Na obrázku 7.1 je náhled prostředí pro správu databáze administrátory. Pro přístup do tohoto prostředí slouží odkaz v menu, ve webovém front-end rozhraní.

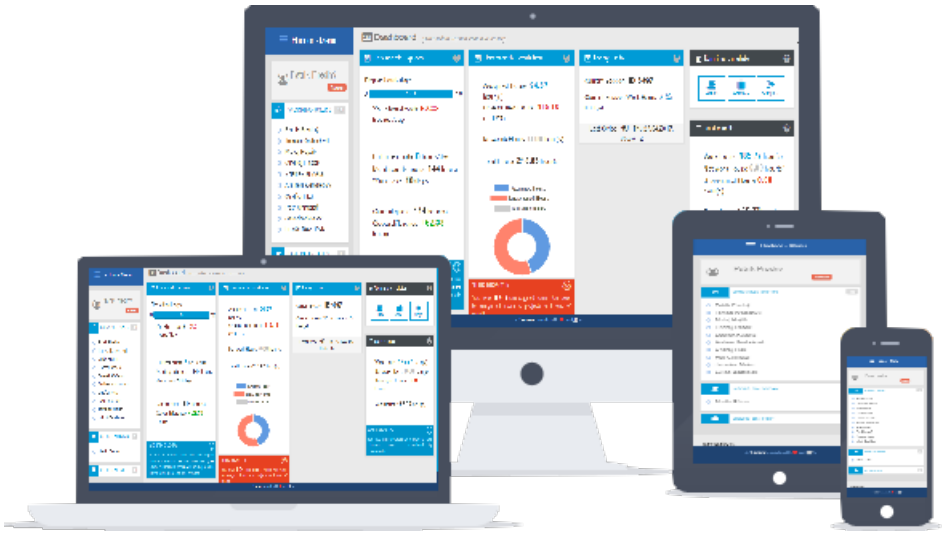


Obr. 7.1: Náhled na grafické prostředí pro správu databáze.

7.3 Hodoor front end

Tato podkapitola popisuje uživatelské front-end rozhraní pro uživatele Hodoor-u, docházkového systému. Je úzce spjaté s Django Back end-em, přímo je integrované ve složce *templates*. Využívá šablony pro vykreslování obsahu a statické soubory pro nastavení designových stylů, JavaScriptových funkcí a podobně.

Uživatel po zadání domény docházkového systému do prohlížeče, vnímá toto rozhraní jako klasickou webovou stránku. Obsahuje uživatelskou sekci tzv. nástěnku (dashboard, obrázek 7.7), správu swipe-ů (obrázek 7.11), sessions stránku (obrázek 7.12) a administrátorskou sekci (obrázek 7.13), sloužící pro kontrolu zaměstnanců. Dále umožňuje uživatelům sledovat svůj odpracovaný čas a přiřazovat jej k projektům. Je to prakticky malý, ale uživatelsky přívětivý, informační systém docházky.



Obr. 7.2: Ukázka responsivního designu Hodoor.

Uživatelské front-end rozhraní je založeno na technologiích HTML/CSS za použití frameworku Bootstrap 3 verze (kapitola 3). Výhodou je právě zmiňovaný responsivní design stránek a aplikací, kdy postačí psát jeden kód jak pro PC, tak pro mobilní zařízení.

Obrázek 7.2 poukazuje na responsivitu designu front end-u Hodoor. Vlevo notebook, uprostřed monitor klasického počítače, vpravo pak tablet a smartphone. Rozměry se automaticky přizpůsobí dle použitého zobrazovacího zařízení.

Styly jsou „vylepšeny“o mnohé funkce díky použití Sass (SCSS) jazyka, který je nadstavbou čistého CSS. Právě Sass dovoluje použití různých podpůrných funkcí, jako jsou proměnné, slučování, zaobalování a další [34].

7.3.1 Struktura HTML šablon ve front-end rozhraní

Tato podkapitola popisuje hlavní stránku šablon *base.html*. Jedná se o HTML stránku obsahující prvky z *Django Templates*, pro vkládání proměnných, funkcí a celých stavebních bloků. Pomocí bloků jsou vkládány dynamicky generované šablony *user_page.html, administrator.html ...*

Stránka šablony *base.html* obsahuje klasickou HTML strukturu stránky. Má hlavičku a tělo. V hlavičce HTML jsou vloženy skripty a styly, podobně jako u klasických webových stránek. Rozdíl je ve vkládání statických souborů příkazem:

```
{% static '<cesta k souboru>' %}
```

Tyto příkazy vkládají statické soubory, se kterými se dále nepracuje a prohlížeč si je ukládá do svojí vyrovnávací paměti. Tento proces následně urychluje načítání dat a celkově zrychluje renderování výsledné struktury.

Dále je možné generování a konverze SCSS souborů na CSS, nebo JS souborů, pomocí příkazu:

```
{% compress <typ souboru> %}
<cesta k souboru>
{% endcompress %}
```

Je ale potřebné mít nainstalovaný modul **Compress** pro Django. Ten překonvertuje soubory a po znovunačtení stránky v prohlížeči aktualizuje vzniklé statické soubory CSS/JS a další.

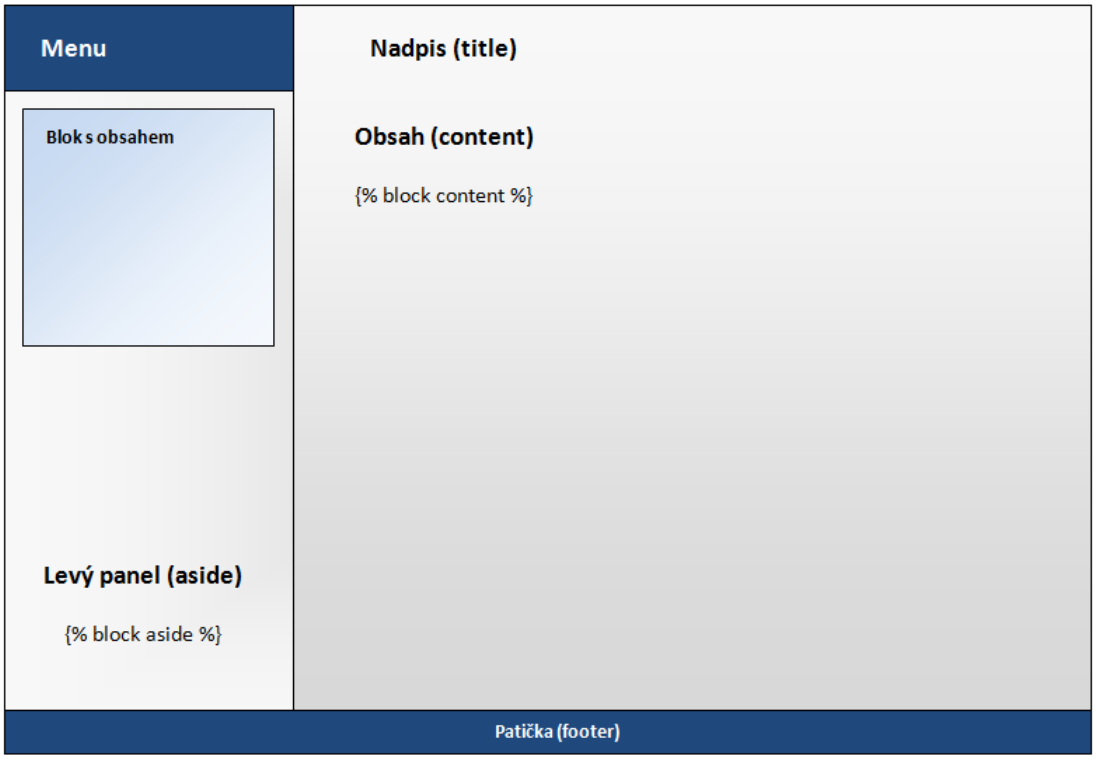
Je to vhodné zejména pro vývoj, a nutné pro práci se Sass preprocesorem (SCSS). Kód pro vkládání dalších šablon do šablony base.html pomocí bloků **Django Templates** je následující:

```
<!-- aside content here-->
        {% block aside %} {% endblock %}
<!--//aside content here-->

<!-- content here-->
        {% block content %} {% endblock %}
<!--//content here-->
```

Blok *aside* představuje levý informační panel, a blok *content* obsahuje zase hlavní obsah stránek. Z dalších šablon stačí potom vložit mezi tyto bloky HTML obsah a ten se automaticky vyrenderuje do šablony base.html.

Kód se vloží právě do místa, které je ohraničené syntaxí **{% block ... %}** a **{% endblock %}**. Takto lze libovolně vkládat další bloky šablon.



Obr. 7.3: Navržená struktura šablony base.html.

Obrázek 7.3 znázorňuje vizualizaci HTML tagů představující bloky, dle kterých se renderuje obsah stránek a podstránek (šablon). Stránka je strukturovaná na levý a pravý sloupec. V levém (aside) se nachází menu podstránek, informační a nastavovací celky. V pravém je vyrenderován obsah aktivované stránky. Obsah (content) obsahuje také nadpisy (title), samotný text, tabulky a další. Posledním prvkem je patička stránky (footer). Tato navržená struktura je plně kompatibilní s požadavky na validní strukturu standardu HTML5 [26] .

Každá z níže uvedených šablon, obsahuje mimo HTML obsahu také vlastní JS funkce. Ty jsou zapisovány mezi syntaxi HTML tagů *<script>* a *</script>*. Jedná se zejména o jQuery funkce upravující vzhled, přebíraní dat pro grafy a animace pro danou stránku. Samotná hlavní šablona *base.html* tyto funkce neobsahuje. Nahrává si je z podřazených šablon.

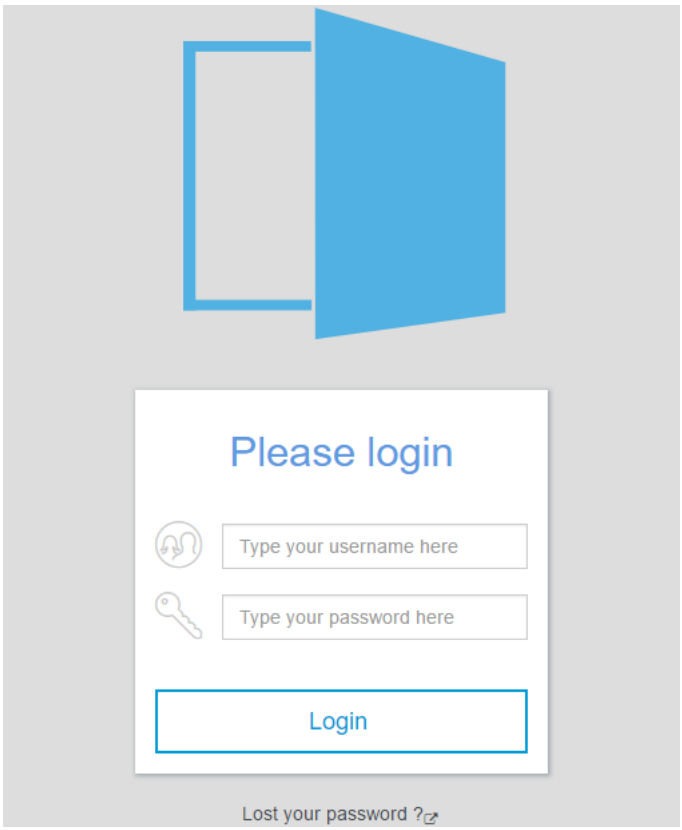
Opakující se funkce, tj. společné pro více šablon, jsou obsaženy v souboru functions.js, a načítají se přímo v hlavičce *base.html*. Šablona *registration/base.html*, má podobné prvky jako šablona *base.html* ze složky *attendance*.

Rozdíl je v uspořádaní těla struktury. Šablona *base.html* ze složky *registration*, slouží pro renderování šablon ze stejné složky. Jedná se o stránky pro přihlášení a

změnu hesla uživatele.

7.3.2 Šablona login.html a password-reset

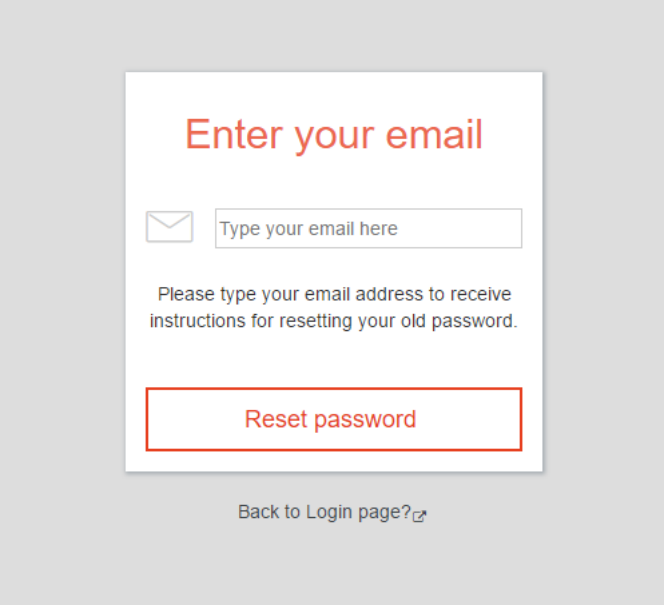
Šablona ze souboru *login.html* je základní šablonou zobrazující se nepřihlášeným uživatelům hned po zadání domény rozhraní. Má podobu klasického formuláře pro zadání jména a hesla, obohaceného o animace a logo docházkového systému Hodoor (obrázek 7.4).



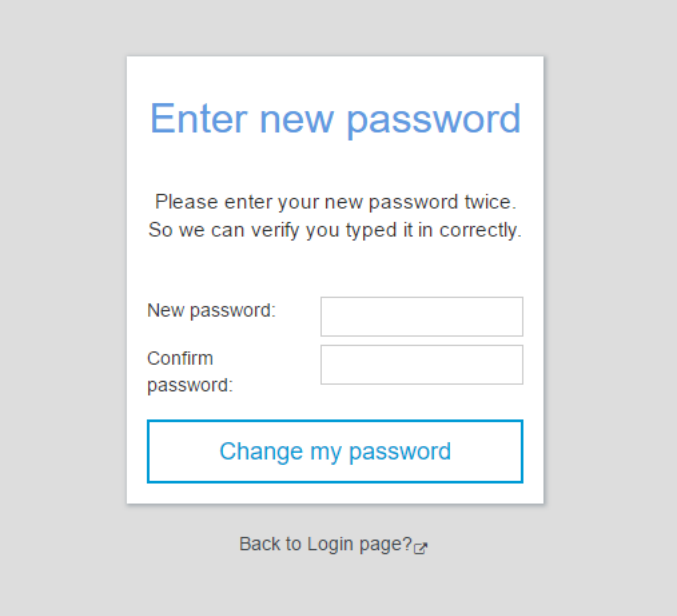
Obr. 7.4: Šablona pro přihlášení uživatel (login.html).

Šablony souborů *password-reset-<funkce>.html*, slouží pro obnovení ztraceného hesla. Heslo se odešle na emailovou adresu zadanou uživatelem (obrázek 7.5). Po obdržení emailu je uživatel naveden na odkaz stránky pro zadání nového hesla (obrázek 7.6).

Jedná se o redundanci, která je v dnešní době již samozřejmostí. Změna hesla by jinak vedla k zásahu administrátorů a ubírala by jeho časové kapacity. Takto jsou uživatelé oprávněni ke změně svého hesla sami a systém je o to víc autonomní.



Obr. 7.5: Šablona password-reset-email.html.

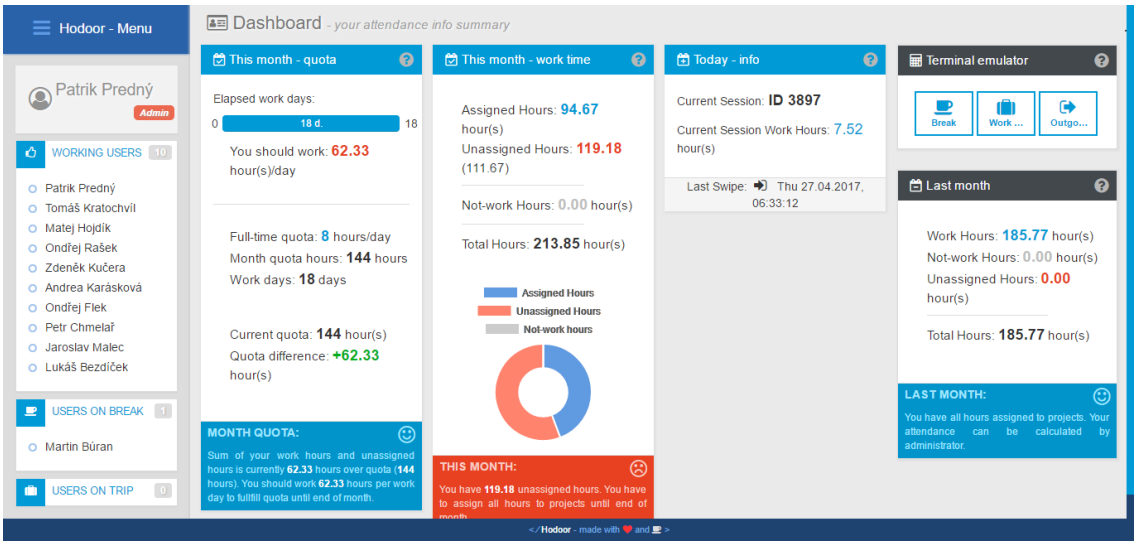


Obr. 7.6: Šablona password-reset-form.html.

7.3.3 Šablona user_page.html

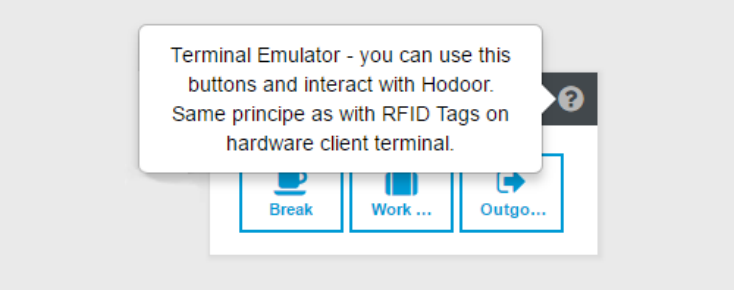
Stránka je primární stránkou zobrazující se uživatelům po přihlášení. Slouží jako úvodní „nástěnka“ s informacemi o docházce. Zobrazuje přehledně v blocích docházku z aktuálního a minulého měsíce.

Také obsahuje informaci o aktuální session, jejím trvání a kvótu pro daný měsíc. V neposlední řade obsahuje emulátor klientského terminálu, kde uživatel může pohodlně z webového prostředí zadávat swipe-y.



Obr. 7.7: Uživatelská nástěnka (user_page.html) šablona.

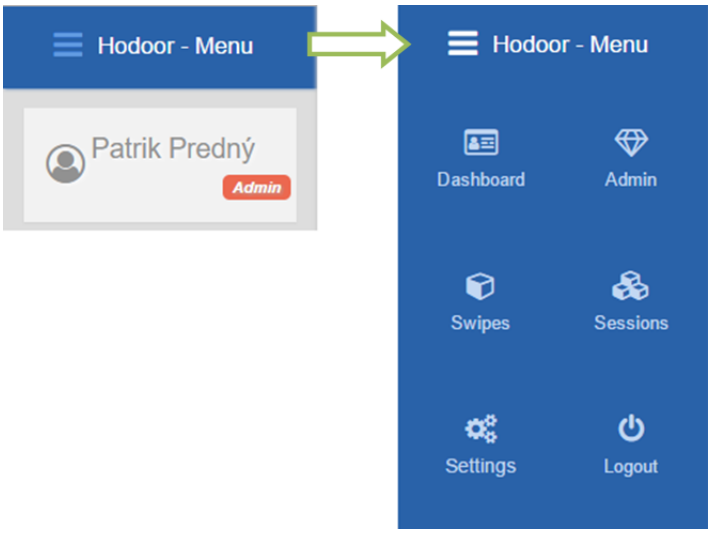
Jednou ze zajímavých funkcí prostředí jsou nápovědy. Ty se zobrazují v přehledných vyskakovacích bublinách, jakou představuje obrázek 7.8. Nápovědy se zobrazují vždy po najetí myší nad ikonu otazníku.



Obr. 7.8: Představení funkce nápovědí.

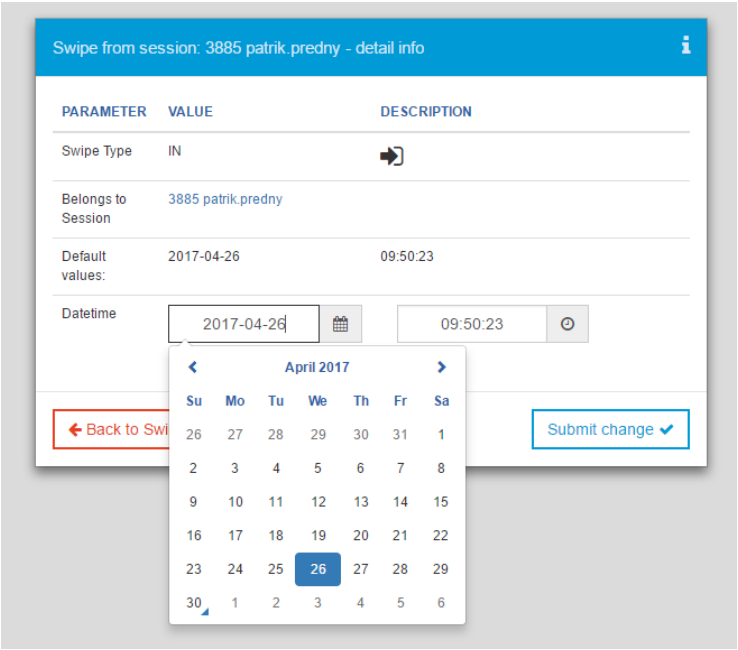
Obrázek 7.9 představuje náhled na komponentu menu. Menu obsahuje odkazy pro přístup k dalším podstránkám prostředí. Obrázek zleva představuje zasunuté

menu, po kliknutí myší se provede jQuery animace a menu se vysune pohybem směrem dolů.



Obr. 7.9: Komponenta menu (vlevo - zasunuté, vpravo - vysunuté menu).

7.3.4 Šablona swipes.html a swipe-detail.html



Obr. 7.10: Swipes-detail šablona - ukázka komponenty pro výběr data.

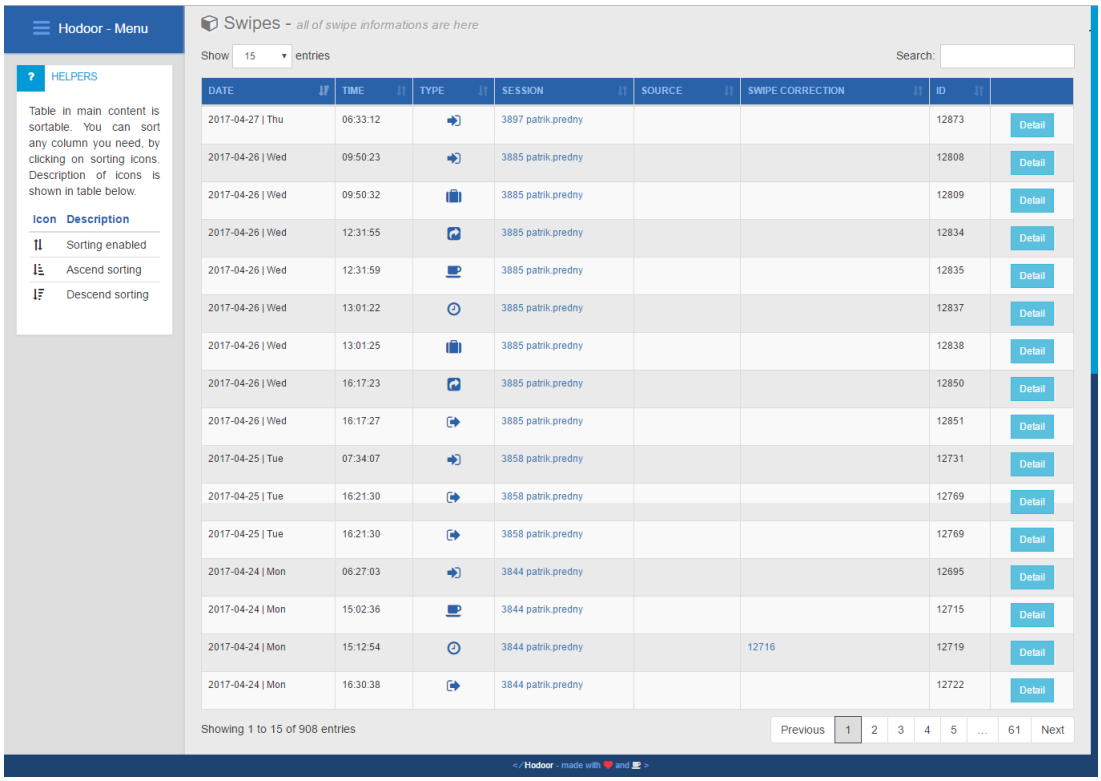
Stránka *swipes.html* je nadřazenou stránkou (obrázek 7.11), stránce *swipes-detail.html*. Swipe-y jsou zde rozloženy v přehledné tabulce, každý z nich má:

- Přiřazený datum a čas vytvoření.
- Typ swipe-u v podobě obrázku.
- Session ID, ke které je přiřazený.
- V případě korekce, zobrazí i korekci.
- Svoje vlastní ID číslo.
- Odkaz na detail.

Na stránku *swipes-detail.html* (obrázek 7.10) se dá přesunout pomocí kliknutí na detail daného swipe-u.

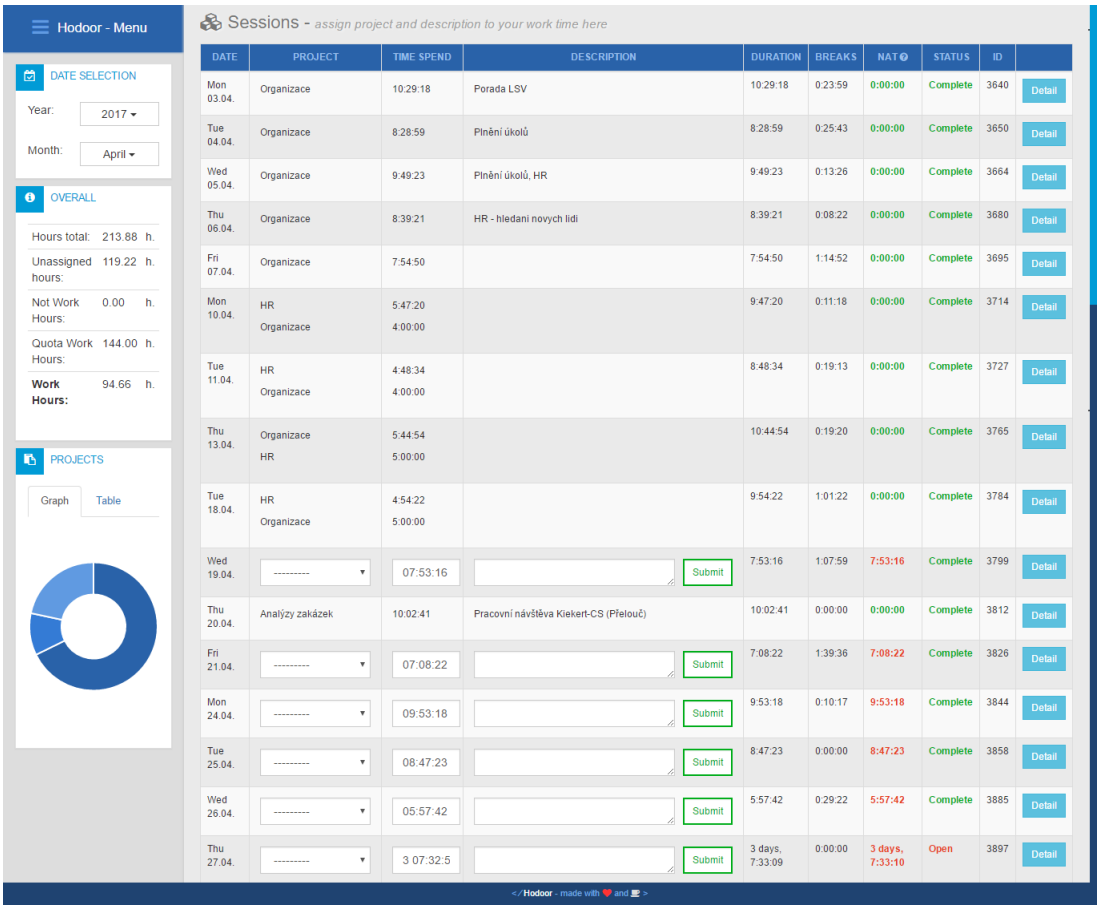
Stránka *swipes-detail.html* nabízí informace o daném swipe-u, který jsme si vybrali na předešlé stránce. Jedná se o modální okno s parametry swipe-u a s možností úpravy data a času pořízení swipe-u. Úprava se projeví jako korekce swipe-u.

Pro lepší UX slouží pro výběr data a času komponenty jednoduchého výběru. S nimi uživatel nemusí údaje zadávat pomocí klávesnice, ale pohodlně je „naklikat“ myší.



Obr. 7.11: Swipes šablona.

7.3.5 Šablona sessions.html a session-detail.html



Obr. 7.12: Sessions stránka (sessions.html šablona).

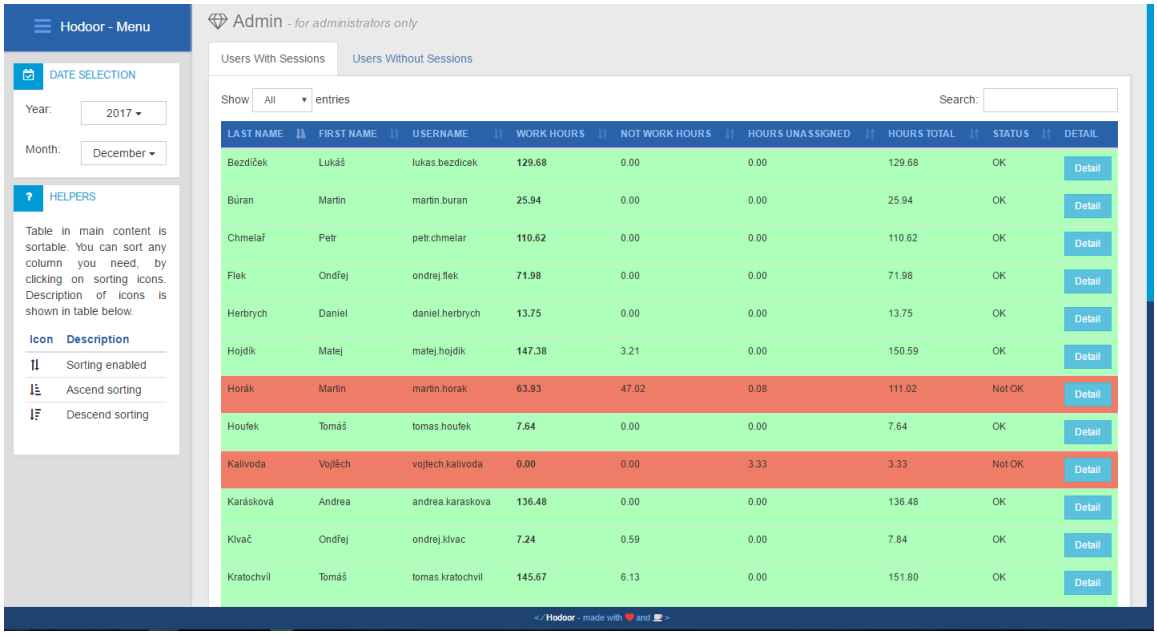
Tato stránka je z pohledu uživatelů přistupujících k docházkovému systému velice důležitá. Zde si každý uživatel vyplňuje výkaz práce za daný měsíc. Ke každé session si přiřazuje projekty a popisy prací, které v daný čas vykonával. Následně po vyplnění informací se řádek tabulky označí jako **Complete**. Pokud jsou všechny řádky tabulky označeny jako hotové (Complete), jsou povinnosti uživatele, z pohledu vyplnění docházky, splněny. Toto je jediná uživatelská povinnost. Jinak čas není přiřazen do hodnoty **Work Time**.

Ve společnosti ELEDUS [2] se nesplnění docházky řeší individuálně. Pro zaměstnance na hlavní, nebo zkrácený pracovní poměr platí obecná pravidla vyplývající ze smlouvy. Proto se například nesplnění měsíční pracovní docházky řeší odčítáním z dovolené a podobně. To však není součástí systému Hodoor. Proto si každý uživatel, který bude systém používat nastaví svoje vlastní pravidla vůči zaměstnancům.

[2] Zadavatel diplomové práce

7.3.6 Šablona administrator.html

Administrátorská stránka je dostupná jedině pro uživatele se statusem **Superuser** v Django administraci (obrázek 7.1). Tady lze vidět všechny uživatele a jejich odpracovaný čas za daný měsíc. Lze také vidět data z měsíců z minulosti. Obsahuje zejména tabulku s informacemi o uživatelích a jejich odpracovaném čase.



Obr. 7.13: Administrátorská stránka.

Status uživatele **OK** nebo **Not OK**, představuje informace o tom, zda uživatel přiřadil svůj pracovní čas k projektům, ve stránce *sessions*. Tato informace je navíc zvýrazněná barevně.

Pomocí odkazů **Detail** se lze přesunout na danou session uživatele pro konkrétní měsíc. Měsíce a roky, ve kterých chceme data listovat, je možné zvolit z nabídky v levém menu.

Uživatelé, kteří nemají žádnou session, jsou zahrnuty do další tabulky. Tuto tabulku je možné zobrazit kliknutím na záložku **Users without sessions**, na horní liště. Jedná se převážně o studenty, kteří v daném měsíci nenavštěvovali společnost a neplnili tím docházku.

7.3.7 Balíky modulů NodeJS

Pro účely lepšího udržování kódu je vhodné používat automatizovaný systém distribuce balíčků jakou je i NPM (Node Package Manager). Tento manažér aplikačních balíčků automaticky nainstaluje požadované balíčky a jejich verze, které jsou předem definované v souboru *package.json* v **root** složce projektu Hodoor. Jeho použití je velice jednoduché, soubor package.json lze vytvořit příkazem:

```
npm init
```

Průvodce instalací nás přes rozhraní Windows terminálu navádí pro správné dokončení a nastavení balíků. Samotná instalace potřebných balíčků využívaných v Hodoor front end-u se provede příkazem:

```
npm install <balík> --save
```

Ten je následně uložen i s použitou verzí do souboru *package.json*. Projekt Hodoor front end využívá konkrétně tyto balíčky JavaScriptových modulů:

- *bootstrap* - Bootstrap CSS/JS framework.
- *chart.js* - Vykreslování grafů.
- *datatables.net* - Práce s tabulkami.
- *datatables.net-bs* - Tabulky pro Bootstrap.
- *eonasdan-bootstrap-datetimepicker* - Lepší práce s datem a časem.
- *font-awesome* - Grafické písmo.
- *jquery* - JavaScript-ová knihovna jQuery.
- *moment* - Práce s časem.
- *moment-timezone* - Doplněk pro Moment.js.

Pro balíčky se v root složce s projektem vytvoří složka s názvem **node_modules**, kde jsou uchovány všechny potřebné soubory pro front-end rozhraní tzv. *dependencies*. Aby dokázal front end správně pracovat s těmito balíčky a jejich daty, je potřebné upravit soubor views.py z back-end rozhraní. Pak se při provedení příkazu *collectstatic* nahrávají tyto balíky přímo do *static* složky. Jejíž funkci lépe popisuje odstavec 7.3.9.

7.3.8 Popis a struktura souborů

Soubory pro front-end uživatelské rozhraní mají následující strukturu.

```
attendance
├── static ..... Složka se statickými soubory
│   ├── css ..... Kaskádové styly pro grafiku
│   │   ├── main.scss ..... Hlavní soubor stylů
│   │   ├── colors-sass.scss ..... Nastavení barev
│   │   └── responsive.scss ..... Pro účely responsibility
│   ├── img ..... Složka s obrázky
│   │   ├── favicon ..... Složka s tzv. favoritní ikonkou
│   │   └── door.svg ..... Obrázek dvěří - logo Hodoor
│   ├── js ..... Složka s JS soubory
│   │   ├── functions.js ..... Hlavní soubor funkcí
│   │   └── tooltips.js ..... Obsahuje funkce pro nápovědu
│   └── modules ..... Obsahuje celé moduly použité pro front end
│       └── mobirise ..... Modul s obrázkovým písmem Mobirise
│           ├── style.css ..... Hlavní soubor se stylem mobirise
│           └── fonts ..... Definice písma mobirise
└── templates ..... Django šablony
    ├── attendance ..... Složka šablony docházkového prostředí
    │   ├── administrator.html ..... Stránka šablony administrátorské sekce
    │   ├── base.html ..... Hlavní stránka pro generování šablon
    │   ├── session_detail.html ..... Stránka šablony Session detailů
    │   ├── sessions.html ..... Stránka šablony Sessions
    │   ├── swipe_detail.html ..... Stránka šablony Swipe detailů
    │   ├── swipes.html ..... Stránka šablony Swipes
    │   └── user_page.html ..... Stránka šablony uživatelské nástěnky
    └── registration ..... Složka se šablonami přihlášení
        ├── base.html ..... Hlavní stránka pro generování šablon
        ├── invitation_email.html ..... Šablona stránky uvítacího emailu
        ├── login.html ..... Šablona stránky pro přihlášení uživatel
        ├── password_reset_complete.html .. Šablona stránky pro obnovení hesla
        ├── password_reset_confirm.html
        ├── password_reset_done.html
        ├── password_reset_email.html
        └── password_reset_form.html
```

7.3.9 Spuštění Hodoor

Pro spuštění, testování, ladění a samotné používání, je potřeba zadat do příkazové řádky několik příkazů. Tento odstavec popisuje sekvenci příkazů pro operační systém Windows. K úkonům budeme potřebovat:

- Nainstalovaný Git.
- Nainstalovaný Python 3.4+.

  - Nainstalovaný Pip (Python Package Index - instalátor balíčků pro Python).
- Nainstalované NodeJS.
- Textový editor např.: Notepad++.

Než se pustíme do instalace Hodoor, nainstalujeme Pip:

```
python get-pip.py
```

Prvním krokem, k práci na Hodoor, je stažení nejnovější verze Hodoor ze zdroje, v tomto případě se jedná o Github.com, na kterém je docházkový systém udržovaný. Naklonujeme si tedy repositář projektu:

```
git clone https://github.com/hodooor/Hodoor.git
```

Dále vytvoříme minimálně složku pro databázi. Například v cmd, ve Windows:

```
mkdir database
```

Poté, lze zadávat python příkazy. Prvním příkazem je vytvoření cache složky pro statické soubory (popsány v odstavci 7.2.1). Vytvoříme ji přikazem:

```
python manage.py collectstatic
```

V případě, že Hodoor není náš jediný Python-ovský projekt, je zapotřebí mít nainstalováno i **Virtual Enviroment** [45]. Struktura souborů v projektu je potom následující:

Projektová složka
- **Hodoor**
- **virtualenv**
- **database**
- **static**

Zadáním sledu následujících přikazů doinstalujeme veškeré potřebné balíčky (dependencies) pro back end (python pip) a pro front end (nodejs npm).

```
cd Hodoor
pip install -r requirements.txt
npm install
python manage.py migrate
python manage.py runserver
```

Migrují se soubory a spustí server. Ten se spouští primárně na adrese **http://localhost:8000/**. Přístupný je samozřejmě z webového prohlížeče např. *Google Chrome*. Následně je nutné vytvořit v databázi prvního uživatele s administrátorskými právy:

```
python manage.py createsuperuser
```

Tímto je nastavování u konce a lze začít pracovat s docházkovým systémem Hodoor.

8 KLIENTSKÝ TERMINÁL

Tato kapitola popisuje mimo potřebný materiál k sestrojení klientského terminálu, také návod na jeho sestrojení a uvedení do provozu. Obsahuje zejména návod a sled unix/linux příkazů pro instalaci a nastavení systému. Dále se věnuje zprovoznění prostředí Electron, ve kterém bude běžet samotná aplikace terminálu.

8.1 Fyzické sestrojení

Jak již bylo v práci zmíněno, klientský terminál je postavený na platformě Raspberry Pi. Níže jsou popsány potřebné komponenty a sled úkonů pro sestrojení samotného hardware terminálu.

8.1.1 Rozpiska materiálu

Komponenty pro sestrojení klientského terminálu jsou převážně tyto:

- Raspberry Pi 3 model B (ev. nižší verze Raspberry Pi, poté je nutné připojit externí WLAN transceiver).
- 7-palcový LCD dotykový display pro Raspberry Pi (s propojovacími kabely, které jsou součást balení).
- Paměťová karta MicroSDHC (alespoň 8GB, v tomto případě se používá 16GB Class 10 UHC).
- Čtečka RFID připojitelná přes USB s funkcí emulací klávesnice.
- Napájecí zdroj z 230VAC na 5V @ 3A DC.
- 2x kvalitní microUSB napájecí kabely.

Dále byly pro sestavení použity tyto dodatečné komponenty:

- 2x DC napájecí propojovací kabely s protikusy.
- 1x USB prodlužovací kabel (vyveden na montážní krabičku, pro připojení USB klávesnice pro případný debugging[1]).
- Montážní plastová krabička (128 x 258 x 47 mm) z ABS plastu.
- Napájecí konektor DC Jack a protikus na kabel.
- 4x distanční sloupek kovový M3x40 + 4x šrouby M3x15.
- 7x montážní šroubky pro zesilovač[2] M3x15.
- Jack-Jack 3,5mm propojovací (line-in) kabel.

[1] Výraz pro řešení chyb v kódu, hardware...

[2] Byl použit modul audio zesilovače fy. ELEDUS. S výstupním výkonem 2W

- Reproduktor do výkonu 2W.

Tab. 8.1: Odhad cen za jednotlivé komponenty

Název komponenty	**Distributor**	**Cena**
Raspberry Pi 7 palcový display	Farnell	1 500 Kč
Raspberry Pi 3 model B	Farnell	1080 Kč
SD karta - SDHC Micro SD Card	Czc.cz	140 Kč
Montážní krabička	GES	86 Kč
RFID čtecka	Ebay	150 Kč
Konektor modular DC Jack + protikus	GES	38 Kč
Audio zesilovač (vhodná náhrada za ELEDUS modul)	Ebay	25 Kč
2x MicroUSB kabel	Farnell	105 Kč
Line in - Jack-Jack 3.5mm - samec-samec	GME	30 Kč
RTC modul DS3231	Ebay	20 Kč
USB 2.0 kabel propojovací (samec - samice)	GME	30 Kč
Celková cena		**3204 Kč**

Dle tabulky 8.1 a s uvážením nulových nákladů za software docházkového systému pro další uživatele, v porovnání s odhadem ceny za standardní komerční řešení (viz. kapitola 1, podkapitola 1.2), lze usoudit, že Hodoor - docházkový systém je nenákladnou variantou jednoduchého docházkového systému. Navíc tento systém má obrovskou výhodu v jeho modulárnosti a možnosti dalšího bezproblémového rozšiřování, dle požadavků dalších uživatelů.

8.1.2 Postup pro sestrojení terminálu

Tato podkapitola popisuje jeden z mnoha způsobů sestrojení terminálu. Konkrétní níže popsané postupy byly použity pro sestrojení terminálu pro testování k diplomové práci. Ve výsledku, si každý další uživatel, může sestrojit klientský terminál dle svého uvážení bez ohledu na tento návod. Musí však obsahovat komponenty popsané v 8.1.1, část I.

Jeden z možných postupů sestrojení a námi použitý (vycházející z obrázku 5.3), je následující:

1. Logicky rozvrhneme komponenty v rámci krabičky.
2. Změříme si prostor pro osazení displaye.

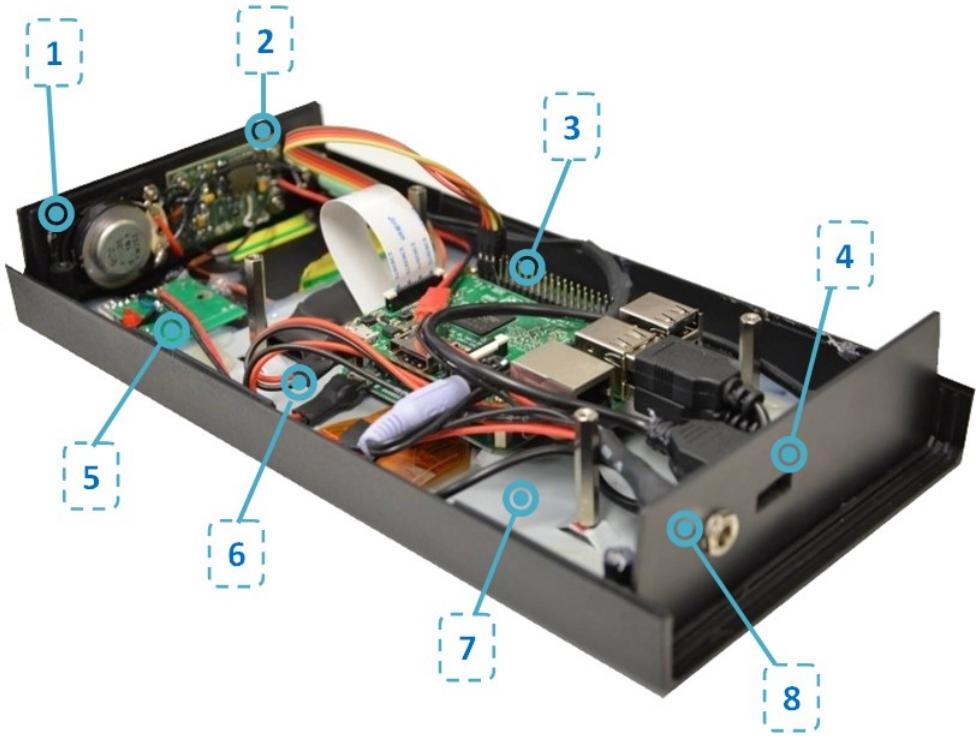
3. Vyřízneme díru pro display.
4. Změříme si pozice montážních otvorů pro uchycení displaye (zadní strana krabičky).
5. Vyvrtáme otvory pro šroubky M3, zašroubujeme distanční sloupky do displaye.
6. Vypilujeme dírku pro hranatou LED RFID čtečky.
7. Vyvrtáme dírky v místě pod reproduktorem.
8. Namontujeme RPi na zadní stranu displaye a zapojíme veškeré konektory.
9. Na boční stěny uchytíme reproduktor se zesilovačem, dále vyvrtáme otvor pro DC Jack a vypilujeme otvor pro USB konektor.
10. Na modul audio zesilovače, RFID čtečku napájíme napájecí kabely, propojíme jack-jack 3,5mm s RPi.
11. Na DC Jack napájíme 2x Micro USB kabely (napájí RPi a display) a protikusy k napájení zesilovače a RFID čtečky (všechny komponenty na 5V DC).
12. Osadíme zařízení na boční stěny a přilepíme USB prodlužovací konektor ke stěně s DC jack přívodním konektorem.
13. Přilepíme RFID čtečku ke krabici terminálu.
14. Sestavíme všechny komponenty krabice a přišroubujeme k sobě.

8.1.3 Fotodokumentace k terminálu

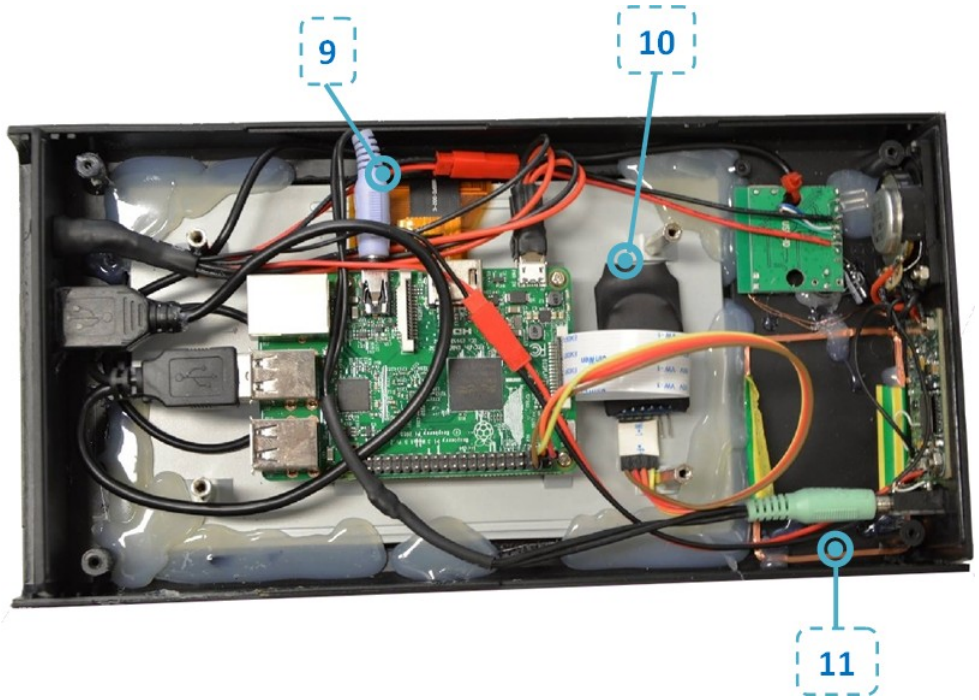
Obrázky 8.1 a 8.2 společně zobrazují pohled na jednotlivé komponenty a jejich možné rozložení v rámci montážní krabice terminálu. K obrázkům je popis uveden níže.

Dále obrázek 8.3 znázorňuje pohled na zkompletovaný terminál, kde je vidět 7-palcový display a nálepku RFID. Tato nálepka slouží pouze pro navedení uživatele k místu možné interakce s RFID tagem a čtečkou. Nemá jiný další význam, avšak je v tomto případě velice důležitou součástí terminálu.

Obrázky 8.4 a 8.5 představují pohledy na boční strany terminálu. Obrázek 8.4 - na boční straně je vyveden USB vstupně-výstupní konektor a konektor DC Jack pro přivedení vstupního napájecího napětí 5V DC do terminálu. Obrázek 8.5 - pravá strana terminálu je osazena audio zesilovačem a reproduktorem. Na obrázku jsou vidět dírky pro přenos zvuku z nitra terminálu.



Obr. 8.1: Rozložení komponent terminálu, část. 1.



Obr. 8.2: Rozložení komponent terminálu, část. 2.

Legenda popisů komponent z obrázků 8.1 a 8.2 je následující:

1. Reproduktor s výkonem 2W.
2. Modul audio zesilovače společnosti ELEDUS s výkonem 2W.
3. Raspberry Pi.
4. USB I/O konektor.
5. Modul RFID čtečky.
6. Napájecí rozvody 5V DC.
7. Konektor pro přívod napájecího napětí 5V DC.
8. 7-palcový dotykový display pro Raspberry Pi.
9. Audio rozvody - line-in audio kabel.
10. Modul RTC - hodin přesného času.
11. Anténa RFID čtečky.

Obr. 8.3: Pohled na terminál seshora.

Obr. 8.4: Levá boční strana terminálu.

Obr. 8.5: Pravá boční strana terminálu.

8.2 Instalace operačního systému

V této kapitole je popsán návod na instalaci operačního systému Raspbian (viz. kapitola 2.3). Dále se věnuje prvotnímu nastavení OS a Raspberry Pi.

8.2.1 Získání distribuce a instalace

Existují dvě verze z každé distribuce. S klasickým CLI a grafickým GUI prostředím. Ideální pro zjednodušení, je instalace verze OS s grafickým prostředím, v tomto případě PIXEL prostředí. A to zejména z důvodu použití displaye.

Na této internetové stránce se nachází jak nejnovější tak starší verze vydaných distribucí. Dobrým zvykem je instalace vždy nejaktuálnější verze systému a ten průběžně aktualizovat - tzv. update. Updaty jsou části aplikací, kde jsou již opraveny chyby odhalené po vydání ostré verze. Tento postup je možný u většiny projektů, pokud to neovlivní samotný chod aplikací. Například změna ovladačů a pod.

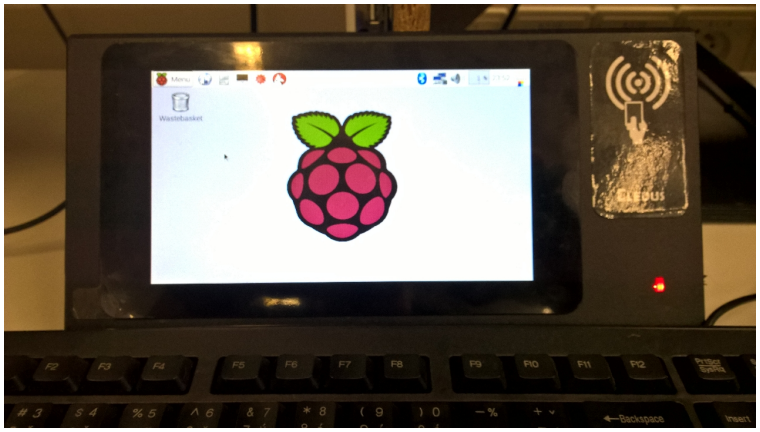
Raspberry Pi si po připojení napájení automaticky načte (bootuje) operační systém z SD karty. Oficiální návod nalezneme na stránkách Raspberry Pi Foundation 22 .

V tomto případě použijeme k vytvoření bootovatelné SD karty počítač se systémem Windows. Postup instalace je následující:

1. Z odkazu v literatuře 24 stáhneme distribuci: Raspbian Jessie with Pixel.
2. Rozbalíme komprimovaný diskový obraz.
3. Získáme tak soubor s příponou „.img“.
4. Stáhneme si Win32DiskImager pro Windows 23.
5. Spustíme Win32DiskImager jako administrátor.

6. Vybereme diskový obraz (img file) z disku počítače.
7. Vybereme písmeno (drive letter) SD karty, kterou chceme použít.
8. Klikneme na tlačítko „Write“.

Tím se vytvoří bootovatelná SD karta ihned použitelná pro aplikování do Raspberry Pi. Po vložení se systém začne automaticky načítávat (bootovat). V případě, že vše proběhlo v pořádku zobrazí se nám na displayi úvodní obrazovka pracovní plochy OS (obrázek 8.6).



Obr. 8.6: Úvodní obrazovka po spuštění Raspbianu na Raspberry Pi.

8.2.2 Instalace modulu RTC

Pro synchronizaci a nastavení času je vhodné použití modulu reálného času (zkráceně RTC). Konkrétně pro Raspberry Pi existuje několik takovýchto modulů. Vybrán byl modul DS3231 připojitelný pomocí sběrnice I2C. Pro jeho zprovoznění je potřeba vykonat několik úkonů. Jsou to zejména:

1. Instalace a zprozovnění I2C komunikace (viz. [41]).
2. Instalace knihovny pro RTC modul (viz. [42]).
3. Nastavení času a jeho zápis do paměti RTC modulu (viz. [42]).
4. Test funkčnosti (viz. [42]).

Prvním krokem je tedy zprovoznění I2C komunikace. Hardwarově je tato sběrnice dostupná, ačkoliv je nutné ji nastavit a používat v systému Raspbian. Nastavení provedeme pomocí příkazu v terminálu Raspbianu. Stisknutím kláves **Ctrl+Alt+t** se otevře okno terminálu a zadáme následující sekvenci:

```
sudo raspi-config
```

Otevře se nabídka s názvem okna: *Raspberry Pi Software Configuration Tool*. Po jejím otevření postupujeme následovně:

1. Klikneme na *Advanced Options*,
2. klik na *A7 I2C* - Enable/Disable...,
3. potvrdíme 2x *YES*,
4. rebootujeme systém příkazem níže.

```
sudo reboot
```

Po restartu systému doinstalujeme potřebný software pro práci s I2C.

```
sudo apt-get update
sudo apt-get install -y python-smbus i2c-tools
```

Vypneme Raspberry Pi:

```
sudo halt
```

Po provedení předchozích kroků, při zapojeném modulu RTC na I2C sběrnici Raspberry Pi, můžeme zkontrolovat funkčnost komunikace.

```
sudo i2cdetect -y 1
```

Terminál vrátí následující:

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

To znamená, že je modul zapojen správně a komunikuje. Jeho adresa je 0x68(hex). Dále přistoupíme ke zprovoznění samotného modulu RTC. Zaktualizujeme software Raspberry Pi:

```
sudo apt-get update
sudo apt-get -y upgrade
```

Zmodifikujeme systémový soubor s moduly:

```
sudo nano /etc/modules
```

Doplníme náš modul:

```
snd-bcm2835
i2c-dev
...
rtc-ds1307
```

Moduly DS1307 a DS3231 používají jedinou knihovnu a to *rtc-ds1307*. Po dokončení uložíme pomocí kláves **Ctrl+X** a **Enter**. Restartujeme systém. Zapíšeme náš přidaný modul do souboru *rc.local*.

```
sudo nano /etc/rc.local
---
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
hwclock -s
```

Vypneme a uložíme (Ctrl+X a Enter), zresetujeme systém. Nastavíme datum, čas a časové pásmo.

```
sudo raspi-config
```

Nebo můžeme použít tento příkaz.

```
sudo date -s "29␣AUG␣1997␣13:00:00"
sudo hwclock -w
```

Příkazem *hwclock -w* se data uloží do paměti RTC modulu. Test funkčnosti se provede odpojením Raspberry Pi od napájení a po určitém čase, bez přístupu na internet se čas a datum musí shodovat s námi nastaveným + prodlevou, kdy bylo Raspberry Pi vypnutné.

8.2.3 Instalace NodeJS

V základní sestavě systému Raspbian není tento balík k dispozici. Jak už bylo zmiňováno v kapitole 3, terminálová aplikace bude založená na webových technologiích (Electron společně s NodeJS). Instalaci a zprovoznění prostředí Electron provedeme pomocí software-u NPM (Node Package Manager). Instalace NodeJS na Raspberry Pi [43]:

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo apt-get install -y build-essential
```

Pokud-li použijeme pro práci s RPi některý z remote-control SW, jakou je např. Putty, musíme provést export obrazovky:

```
export DISPLAY=:0.0
```

Nyní je vše připraveno pro implementaci samotné JS aplikace terminálu (kapitola 9).

9 APLIKACE KLIENTSKÉHO TERMINÁLU

Tato kapitola popisuje již vytvořenou aplikaci klientského terminálu založenou na technologií Electron. Prostředí Electron umožňuje aby byla celá aplikace napsána pomocí webových technologií jako jsou již zmiňované jazyky HTML, CSS a JavaScript (kapitola: 3 odstavec 3.2).

9.1 Prostředí Electronu

Hodoor-ová aplikace klientského terminálu, založená na technologií Electron, pracuje také s knihovnou jQuery. jQuery je knihovna pro usnadnění práce se samotným, dnes už nepružným JavaScriptem. Vyznačuje se převážně předdefinovanými funkcemi, které umožňují například jednoduché animace objektů. Objekty jsou nejčastěji prvky HTML stránky.

9.1.1 Vytvoření holé Electron aplikace

Na portálu Github.com nalezneme odkaz na repositář aktuální verze čisté (holé) electronové aplikace. Příkazem se provede naklonování aplikace do námi zvolené složky v PC:

```
git clone
https://github.com/electron/electron-quick-start.git
hodoor-terminal-app
```

Poté se přesuneme do složky, která se vytvořila po stažení repositáře:

```
cd hodoor-terminal-app
```

Samozřejmě musíme mít funkční příkaz git. Vytvoří se složka s námi uvedeným názvem *hodoor-terminal-app*, ta bude sloužit jako git *working directory* (kapitola 6). Dále je potřeba mít nainstalovaný *NodeJS* s *npm* (Node Package Manager). Instalaci electron-u a spuštění provedeme příkazem:

```
npm install && npm start
```

9.1.2 Původní struktura souborů Electron

Po otevření naší *working directory* máme k dispozici následující strukturu souborů.

Hodoor-terminal-app
- `.gitignore` ..... Neverzované soubory
- `index.html` ..... Základní struktura/kostra aplikace
- `LICENSE.md` ..... Github.com licence
- `main.js` ..... Hlavní soubor pro electron funkce

```
├── package.json ..... Obsahuje informace o electron-ové aplikaci
├── README.md ..... Holý github.com čti mě soubor
├── renderer.js ..... Iniciální soubor vhodný pro renderovací funkce
└── node_modules
```

9.2 Hodoor - terminal app

Popisuje již naprogramovanou aplikaci klientského terminálu Hodoor, od počátku nastavení, až k popisu jednotlivých funkcí.

9.2.1 Instalace balíčků

Pro práci s Hodoorem a jeho terminálem, je potřebné mít nainstalované tyto doplňující NodeJS balíčky:

- *Auto-launch* - pro samospouštění,
- *JQuery* - pro lepší práci s JavaScriptem,
- *Weather-js* - pro získání aktuálního počasí.

Instalaci a uložení provedeme příkazem v cmd Windows:

```
npm install --save <název balíčku>
```

Balíky se po uložení zapíší automaticky do souboru *package.json*. Tento soubor obsahuje mimo jiné např. název naší aplikace, její popis, verzi a pod. Pro další instalaci a spuštění na jiném zařízení nebo v jiné složce (např. při rozdílných verzích Hodoor - temrinal app), stačí zadat příkaz:

```
npm install
```

Doplňující balíky se již instalují ze seznamu *package.json*, které jsme předtím uložili. Tak máme k dispozici veškeré potřebné balíky pro pokračování ve vývoji. Vytvořením instalačního balíku celé aplikace terminálu se tato práce nezabývá.

9.2.2 Upravená struktura souborů

Struktura aplikace terminálu v Electronu je následující:

```
Hodoor-terminal-app
├── .gitignore ..... Neverzované soubory
├── functions.js ..... Hlavní soubor s funkcemi a logikou
├── index.html ..... Základní struktura/kostra aplikace
├── LICENSE.md ..... Github.com licence
├── main.js ..... Hlavní soubor pro electron funkce
├── package.json ..... Obsahuje informace o electron-ové aplikaci
└── README.md ..... Github.com čti mě soubor s instrukcemi
```

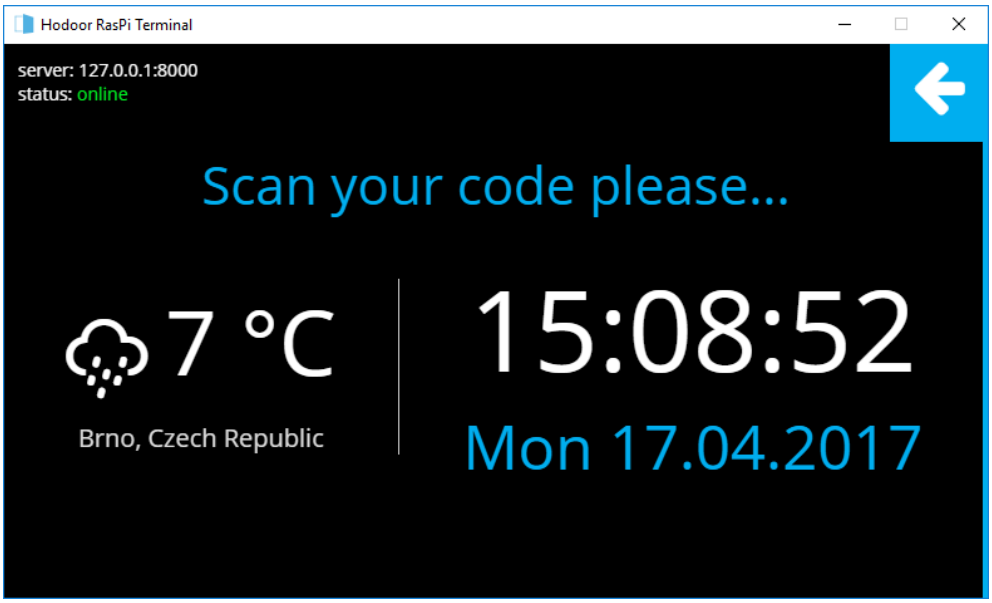
```
settings.json ..... Nastavení terminálu
css
    style.min.css ..... Zkompilovaný SCSS soubor
    style.scss ..... SCSS soubor s grafikou v Sass CSS
    weather-icons-wind.min.css ..... Styl ikon počasí
    weather-icons.min.css ..... Styl ikon počasí
font
    OpenSans-Semibold ..... Použité písmo
    weathericons-regular-webfont ..... Styl ikon počasí
img
    favicon.ico ..... Ikona aplikace
js
    jquery-3.1.1.min.js ..... jQuery knihovna
    render.js ..... Soubor s programem pro vykreslení grafiky
modules
    fontawesome ..... Doplňkové ikony Fontawesome
        css
            font-awesome.min.css
        fonts
            fontawesome-webfont
node_modules ..... Doplňkové balíčky NodeJS
```

9.2.3 Grafický návrh aplikace terminálu

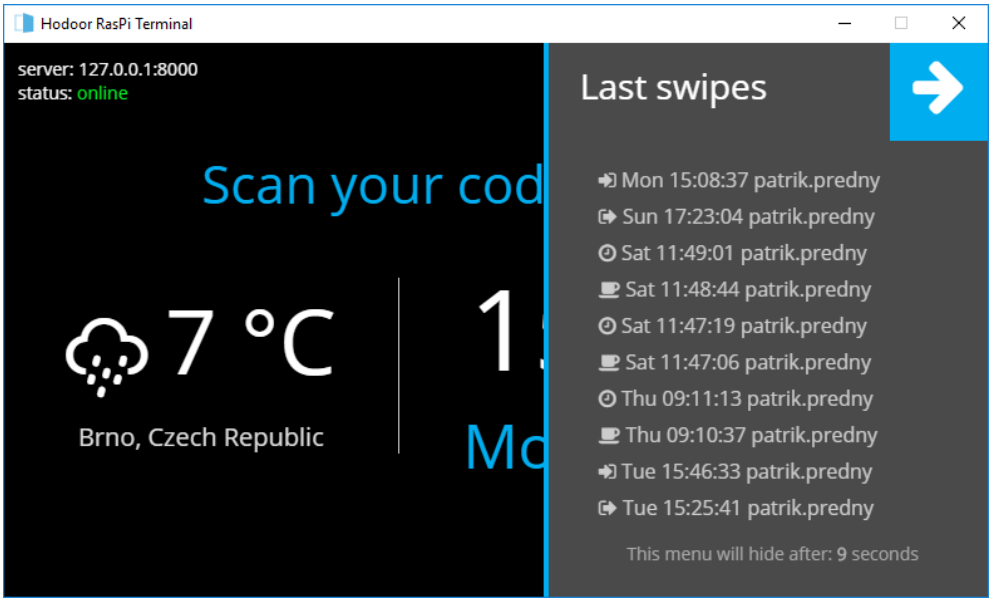
Klientský terminál musí pracovat v několika režimech a to zejména:

- *Server dostupný*
  - *Výpis posledních interakcí* - ukáže poslední interakce uživatelů.
  - *Aktivní režim* - čeká se na interakci s uživatelem.
    - *Klíč zadán správně* - po správném pípnutí RFID čipem, uživatel nalezen v DB serveru a může provádět akci.
      - *Režim odesílání interakce* - po odeslání swipe-u uživatelem musí čekat na přijetí serverem.
    - *Nesprávný klíč* - uživatel nenalezen v DB serveru, terminál vrátí chybovou hlášku.
- *Server nekomunikuje* - server, na kterém běží tzv. back-end pro správu docházky je nedostupný.

Vytvořením požadované struktury z tagů v HTML souboru *index.html* a následném „nastylování“ v souboru *css/style.scss* dostaneme požadovaný vzhled aplikace terminálu (obrázky 9.1 - 9.7).

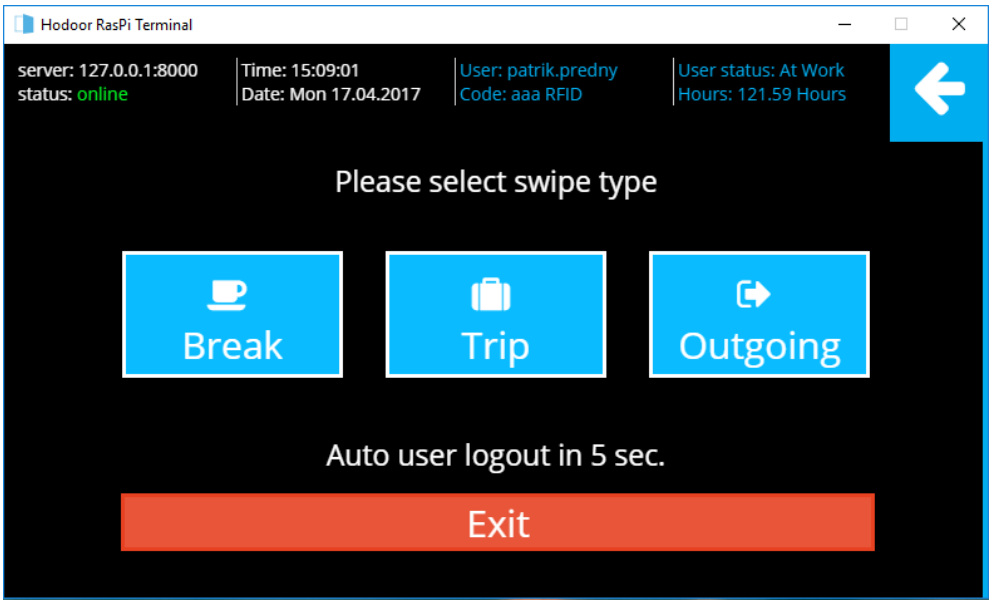


Obr. 9.1: *Server dostupný - Aktivní režim* - čeká se na interakci uživatele.

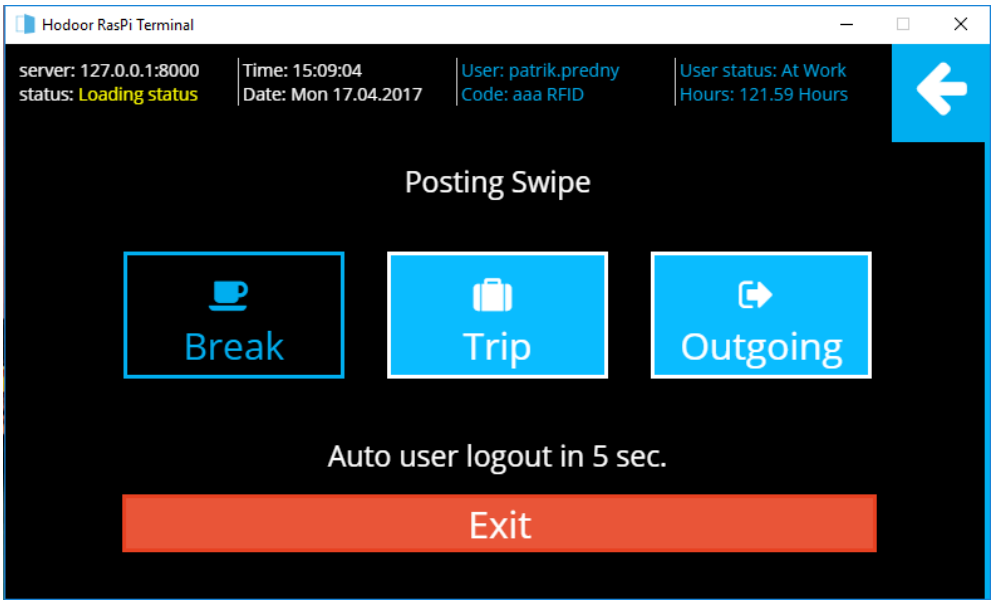


Obr. 9.2: *Server dostupný - Aktivní režim* - výpis posledních interakcí uživatel.

Obrázky 9.1 - 9.7 představují grafický vzhled aplikace klientského terminálu. Každá z nich má svoji specifickou funkci, kterou předává uživateli informace o dění na terminálu. Vzhled je navržen s uvážením nejnovějších trendů v oblasti návrhu - tzv. „flatdesign“, kde se setkáváme se strohými a jednoduchými tvary jednotlivých prvků. Základem každé dobré grafiky je dodržení maximálně tří barev a jejich kombinací.

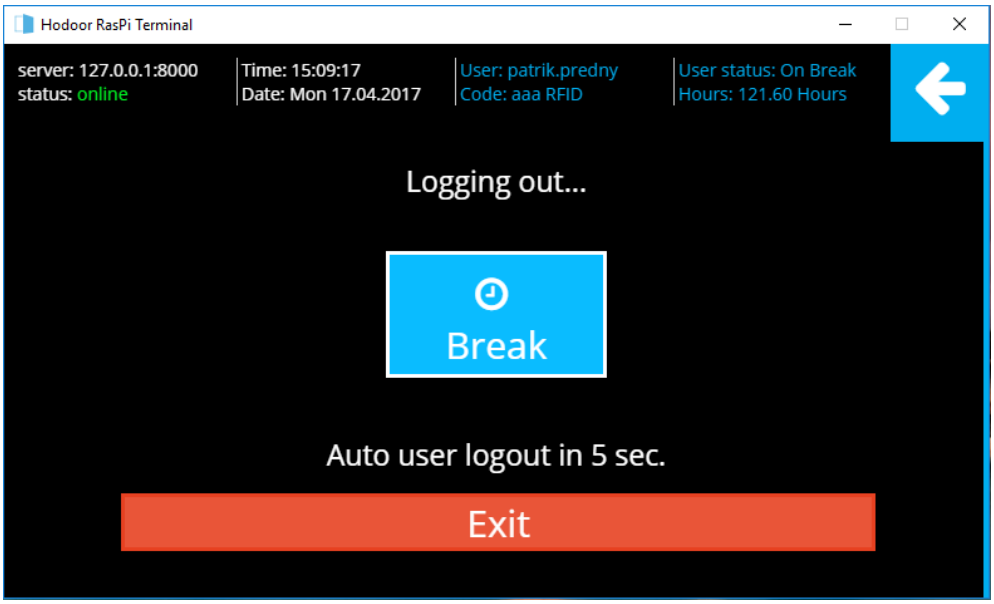


Obr. 9.3: *Server dostupný - Aktivní režim - klíč zadán správně* - nabídka swipe-ů.

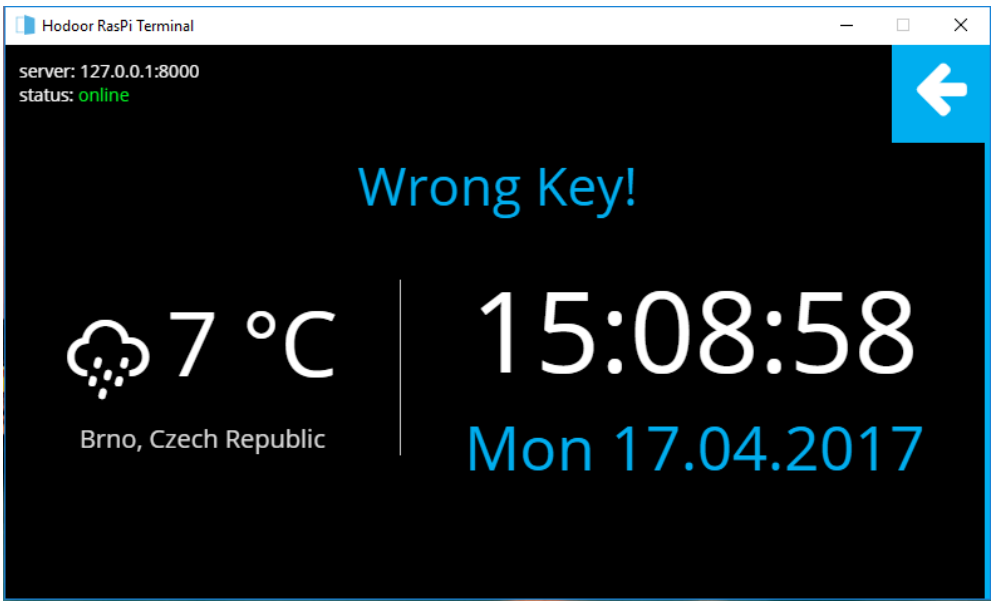


Obr. 9.4: *Server dostupný - Aktivní režim - klíč zadán správně* - odesílání zvoleného swipe-u.

Pro sestavení této aplikace byla použita zadavatelem požadovaná barva (modrá) zadavatelské společnosti. S kombinací černé a bíle tvoří uživatelsky přívětivý vzhled celé aplikace.



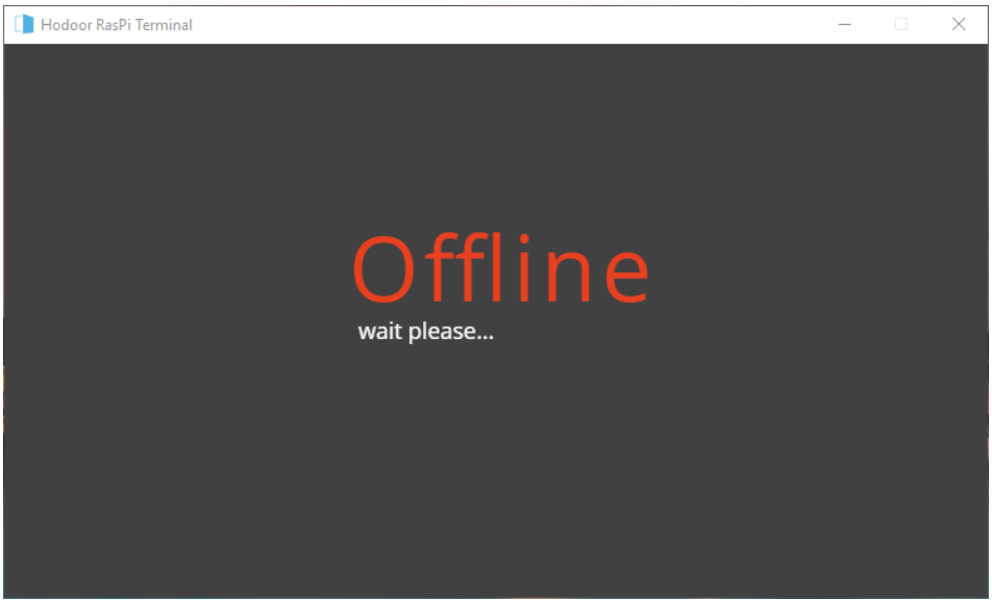
Obr. 9.5: *Server dostupný - Aktivní režim - klíč zadán správně* - obrazovka odhlášení uživatele.



Obr. 9.6: *Server dostupný - Aktivní režim* - nesprávný klíč.

9.2.4 Popis souboru main.js

Soubor *main.js* je hlavním souborem Electron. Obsahuje informace o nastavení prostředí Chromium, jako jsou například velikost vytvořeného okna aplikace, aplikační menu, import dodatečných souborů a jiné. Ukázka kódu ze souboru *main.js* znázor-



Obr. 9.7: *Server nedostupný* - nekomunikuje s terminálem.

ňuje využití balíku Auto-launch.

```
let mainWindow; //definice mainWindow
var AutoLaunch = require('auto-launch'); //načti auto-launch
var appLauncher = new AutoLaunch({
    name: 'Hodoor-client', //pod jm. Hodoor-client
});

appLauncher.isEnabled().then(function(enabled){
    if(enabled) return;
    return appLauncher.enable()
}).then(function(err){ //v případě chyby ji vypíše
        do konzole chromia
    console.log("Error is:" + err)
});
```

Následující funkce definuje vytvořené okno aplikace, jeho rozměry (800x480pixelů), počátek souřadnic, povoluje/zakazuje změnu rozměrů a pod.

```
function createWindow () {
  // Create the browser window.
    if(process.arch == "arm"){
        mainWindow = new BrowserWindow({
            kiosk: true,
            useContentSize: true,
            alwaysOnTop:true
        });
        console.log("We␣are␣on␣Raspberry␣Pi␣:)");
    }
    else{
        mainWindow = new BrowserWindow({
            width: 800,
            height: 480,
            resizable: false,
            icon:'img/favicon.ico',
            x: 0,
            y:0,
            useContentSize: false,
        });
    }

//And load the index.html of the app.
    mainWindow.loadURL('file://' + __dirname + '/index.html');
```

Zajímavou částí je podmínka *if(process.arch == "arm")*, která se řídí architekturou našeho zařízení.

Pokud-li je použita aplikace pro Raspberry Pi, tj. architektura *arm*, místo rozměrů se použije automatická funkce *useContentSize*, nastaví se dle displaye zobrazovacího zařízení a bude nadřazena nad ostatními okny (*alwaysOnTop:true*).

Pro jinou architekturu např. OS Windows, jsou pevně definované rozměry aplikace, ikona, je zakázána změna rozměrů.

Funkce *mainWindow.loadURL...* načte hlavní soubor s HTML strukturou s názvem *index.html*.

9.2.5 Popis struktury HTML

Struktura (kostra) aplikace, která jí dodává fyzickou podobu, je popsána v dokumentu *index.html*. Dokument mimo samotnou strukturu obsahuje i volání funkcí ze

souboru *function.js*. V hlavičce HTML dokumentu se načítají požadované skripty a styly. Je zde nastaveno kódování na UTF-8 a zadán titulek Hodoor Raspi Terminal.

```
<head>
  <meta charset="UTF-8">
  <title>Hodoor RasPi Terminal</title>
  <link rel="stylesheet" type="text/css"
        href="css/style.min.css" />
  <link rel="stylesheet" type="text/css"
        href="modules/fontawesome/css/font-awesome.min.css" />
  <link rel="stylesheet" type="text/css"
        href="css/weather-icons.min.css" />
  <link rel="stylesheet" type="text/css"
        href="css/weather-icons-wind.min.css" />
</head>
```

Tělo HTML struktury, vyhrazeno tagy *<body>* a *</body>*, obsahuje stavební bloky definující jednotlivé obrazovky. Tyto obrazovky jsou vyobrazeny na obrázcích 9.1 a 9.7.

Ku příkladu je níže uvedena struktura obrazovky funkce *SystemCodeScan(), ze souboru render.js*. Postavena z tagů *div* a *span*, grafický vzhled je popsán v souboru *style.scss*.

Tato obrazovka (obr. 9.1) navádí uživatele k přiložení RFID čipu k terminálu. Dále ukazuje serverový čas a aktuální počasí v nastavené lokalitě.

```
<div class="subblock-center">
        <span class="info-text">
                Scan your code please...
        </span>
        <div class="weather-block">
                <span class="icon">
                        <i class='wi wi-day-sunny'></i>
                </span>
                <span class="text"></span>
                <span class="location">
                        Loading weather info...
                </span>
        </div>
        <div class="info-block">
                <span class="time">15:48:05</span>
                <span class="date">Tue 21.03.2017</span>
        </div>
</div>
```

Volání souboru s logikou a dalšími funkcemi na konci HTML kódu:

```
<script>
  require('./functions.js');
</script>
```

9.2.6 Popis funkcí ze souboru render.js

V souboru *js/render.js* jsou zapsány jQuery funkce pro vykreslování grafických obrazovek. Tato podkapitola popisuje jejich funkce vztažené na prvky ze souboru *index.html*.

- **loaderOn** - zakryje element content (obsah) a ukáže loader (animované načítání stránky).
- **loaderOff** - opak funkce loaderOn().
- **updateClock** - aktualizuje hodiny na terminálu s frekvencí 1 sekundy.
- **updateDate** - každou hodinu aktualizuje datum na terminálu.
- **SystemOffline** - ukáže obrazovku z obrázku 9.7.
- **SystemCodeOK** - zobrazí uživateli obrazovku z obrázku 9.3.
- **SystemCodeScan** - zobrazí obrazovku dle obrázku 9.1.
- **ShowLastSwipesMenu** - po kliknutí na šipku z obrázku 9.1 vysune seznam posledních swipe-ů.
- **HideLastSwipesMenu** - při vysunutém swipes seznamu (9.3) po následném kliknutí, nebo po uplynutí určité doby, skryje seznam posledních swipe-ů.
- **btnShow<kombinace swipe-ů>** - tyto funkce zobrazují dle definované logiky swipe-ů (kapitola 5, odstavec 5.2.1) tlačítka pro interakci s uživatelem. Tlačítka slouží k odeslání daného swipe-u na server.
- **resetAll** - uvede texty na obrazovkách do původního stavu. Stav podobný zapnutí a prvotní inicializaci terminálu.

9.2.7 Popis funkcí ze souboru functions.js

Na začátku dokumentu *functions.js* se načítají potřebné moduly. Jsou to:

- *Request* - pro práci s hlavičkou stránek, GET a POST.
- *Settings.json* - JSON dokument s nastavením terminálu. Obsahuje zejména adresu serveru, serverový token a nastavení modulu počasí.
- *render.js* - soubor s funkcemi pro vykreslení grafických prvků (odstavec 9.2.6).
- *jquery* - pro práci s knihovnou jQuery JS.

Kód pro načtení potřebných modulů je následující:

```
const request = require('request');
const settings = require("./settings.json");
const terminal = require('./js/render.js');
const \$ = require('jquery');
```

Popis některých funkcí:

- **keysFromServer** - hlavní volání funkce pro práci s daty ze serveru.
  - **loadAll(loaduser)** - vrátí všechny uživatele a jejich vlastnosti (např. klíče a pípnutí) jako jeden objekt.
  - **loadUser(key)** - načte údaje daného uživatele dle klíče jeho čipu.
  - **getAll** - načte serverovou databázi.
  - **getUser** - vrátí uživatele jako objekt.
  - **unloadUser** - odhlásí uživatele z nabídky swipe-ů.
- **setInterval** - obecná funkce JS, nastaví opakování v určeném čase.
- **keyCodeReader** - pro práci s klíči čipů.
  - **pressed(keycode)** - pokud je zadán klíč čipu, ověří jeho shodu s klíči z DB.
  - **getCurrentKeyCode** - vrátí daný klíč.
- **swipeSender(swipe_type)** - v kladném případě odešle JSON soubor s daty o uživateli a potvrzeném swipe-u. V opačném případě vypíše chybovou hlášku.
- **getLastActionString(shorcut)** - přeloží text posledního swipe-u uživatele do vhodného textového formátu.
- **getSwipeTypeIcon(swipe_type)** - na základě typu swipe-u vrátí vhodnou ikonu přiřazenu ke swipu, tak aby korespondovala s grafickým návrhem.
- **updateSwipeList** - v pravidelném intervalu aktualizuje seznam posledních swipe-ů.
- **getUserName(user_id)** - vrátí jméno uživatele.
- **logOut()** - odhlásí uživatele z obrazovky pro výběr swipe-ů.
- **getWeatherInfo** - pokud-li je modul pro počasí povolen, vrátí aktuální počasí.

9.2.8 Popis souboru settings.json

Tento soubor, formátu JSON, obsahuje nastavení terminálové aplikace.

- *URL* - adresa URL serveru s back-end rozhraním.

- *TOKEN* - komunikační token pro daný server.
- *WEATHER_MODULE_ENABLED* - booleovská konstanta pro povolení/-zákaz modulu počasí.
  - *WEATHER_REFRESH_TIME_SEC* - aktualizační čas dat z modulu počasí.
  - *CITY* - město pro výpis počasí.
  - *COUNTRY* - krajina pro výpis počasí.
  - *UNIT* - jednotka počasí ve °C nebo °F.
  - *AUTO_TERMINAL_LOAD_SEC* - obnovovací čas pro terminál.

9.2.9 Instalace aplikace klientského terminálu

Podobně jako je tomu u čisté Electron aplikace, která je určená pro další vývoj, ty samé úkony provedeme i u instalace **Hodoor-terminal-app** - aplikace klientského terminálu.

```
git clone
   https://github.com/hodooor/Hodoor-terminal-app.git
cd Hodoor-terminal-app
npm instal && npm start
```

Po instalaci a spuštění se zobrazí obrazovka aplikace terminálu z obrázku 9.7, je proto potřebné mít spuštěný webový server (kapitola 7) a v souboru *settings.json* nastavené správné údaje serveru. Je potřebné zadat adresu serveru do řádku **URL** a serverový token, řádek **TOKEN** v souboru *settings.json*. Následně po dalším spuštění se zobrazí obrazovka aplikace z obrázku 9.1.

Instalaci na Raspberry Pi se provede podobným způsobem. Nejprve na Raspberry Pi naklonujeme repositář s naší aplikací a poté příkazy install a start aplikaci spustíme (stejný postup jako u instalace ve Windows).

Je-li to nutné, nastavíme unixový plánovač (tzv. Cron) tak, aby aplikaci spouštěl po startu terminálu.

```
crontab -e

@reboot export DISPLAY=:0 &&
   /home/pi/Hodoor-terminal-app/node_modules/.bin/electron
   /home/pi/Hodoor-terminal-app/main.js
```

Nyní se po každém nabootovaní systému na terminálu spustí také terminálová aplikace.

10 ZÁVĚR

Práce se zaměřuje na open-source projekt s názvem Hodoor. Jedná se o případový docházkový systém (casual attendance system), který zadala firma ELEDUS s.r.o., jako diplomovou práci, dle svých vlastních požadavků. Projekt je uložený v podobě repositáře na úložišti Github.com. Je to projekt s otevřeným kódem, takže si ho může každý zájemce upravit dle svých vlastních požadavků. Práce je pomyslně rozdělena na teoretickou a praktickou část.

Hlavními úkoly bylo sestrojit fyzicky klientský terminál na Raspberry Pi, k němu vytvořit aplikaci pro komunikaci se serverem. Další částí zadání bylo vytvořit uživatelsky přívětivé webové prostředí front-end rozhraní docházkového systému. back-end rozhraní nebylo úlohou této práce. Samotná diplomová práce začíná definicí problematiky, a končí fyzickou realizací a splněním všech požadavků docházkového systému.

Teoretická část diplomové práce je zaměřena především na všeobecné poznatky o technologiích, které jsou využívány pro praktickou část. Kapitola 1 popisuje základní analýzu problémů, 2 popisuje embedded počítač pro realizaci terminálu, 3 obsahuje úvod do moderních webových technologií, na kterých je v praktické části postavena aplikace pro klientský terminál. Z důvodu použití RFID přístupových čipů je v kapitole 4 popsána problematika automatické identifikace a sběru dat, a je vybrána vhodná RFID čtečka pro terminál.

Praktická část práce se dělí celkově na tři hlavní bloky. Jsou to front-end uživatelské rozhraní, klientský terminál založený na Raspberry Pi a aplikace klientského terminálu. Tyto tři stavební bloky jsou dílem autora diplomové práce. Základy aplikace klientského terminálu položil konzultant práce Ing. Ondřej Vičar. Systém potřebuje ke své plné funkci také back-end rozhraní a databázi, nicméně jsou v práci popsány jen okrajově, z důvodu, že nejsou dílem autora této diplomové práce.

Front-end uživatelské rozhraní docházkového systému je grafické prostředí pro přístup uživatel ke své docházce a je popsáno v kapitole 7. Hardwarový klientský terminál založený na platformě počítače Raspberry Pi je popsán v kapitole 8. Zde je uveden kompletní návrh, s výběrem komponent, až po samotnou fyzickou realizaci. Posledním bodem zadání bylo napsat aplikaci pro terminál. Ta je obsažena v kapitole 9 a je založena na technologiích Electron a JQuery.

Celý systém terminálu má sloužit jako low-cost řešení pro jednotlivce, malé firmy, nebo jako startovací projekt pro nadšené vývojáře. Výhodou je pestré spektrum použitých nejmodernějších technologií a rozsáhlá dokumentace kódů. Docházkový systém Hodoor šetří čas i peníze všem jeho uživatelům.

LITERATURA

[1] Wikipedia The Free Encyclopedia, *Raspberry Pi*. [online]. [cit. 2016-10-20]. Dostupné z: https://en.wikipedia.org/wiki/Raspberry_Pi.

[2] RASPI.CZ, *Raspberry Pi 2 - nová turbomalina*. [online]. [cit. 2016-10-20]. Dostupné z: http://www.raspi.cz/2015/02/raspberry-pi-2-nova-turbomalina/.

[3] Opensource.com, *What is Raspberry Pi?*. [online]. [cit. 2016-10-22]. Dostupné z: https://opensource.com/resources/what-raspberry-pi.

[4] Oborový portál pro BOZP, *Evidence docházky*. [online]. [cit. 2016-10-24]. Dostupné z: http://www.bozpinfo.cz/evidence-dochazky.

[5] Docházkové, přístupové a stravovací systémy GACC, *Sestavte si docházkový systém na míru*. [online]. [cit. 2016-10-25]. Dostupné z: http://www.gacc.cz/dochazkovy-system#config.

[6] Wikipedia The Free Encyclopedia, *Automatická identifikace a sběr dat*. [online]. [cit. 2016-10-28]. Dostupné z: https://cs.wikipedia.org/wiki/Automatick%C3%A1_identifikace_a_sb%C4%9Br_dat.

[7] Svět Androida, *Bezdrátové technologie: Co je NFC a jak ho využít?*. [online]. [cit. 2016-10-29]. Dostupné z: https://www.svetandroida.cz/bezdratove-technologie-nfc-201507.

[8] JURÁK K., NEJEZCHLEBOVÁ Z., *RFID, NFC, Bluetooth – Terminologie*, z DPS - Elektronika od A do Z, rok 2014, číslo časopisu DPS 5/2014. [online]. [cit. 2016-11-03]. Dostupné z: http://www.dps-az.cz/zajimavosti/id:12192/rfid-nfc-bluetooth-terminologie.

[9] APIS spol. s r.o., *Čo je biometria?*, [online]. [cit. 2016-11-03]. Dostupné z: http://www.biometria.sk/co-je-biometria.html.

[10] ŠČUREK R., *Biometrické metody identifikace osob v bezpečnostní praxi*, VŠB TU Ostrava, Fakulta bezpečnostního inženýrství, Katedra bezpečnostního managementu, Oddělení bezpečnosti osob a majetku, rok 2008, počet stran 58, [studijní text].

[11] RFID Portál, *Co je RFID*, [online]. [cit. 2016-11-03]. Dostupné z: http://www.rfidportal.cz/index.php?page=rfid_obecne.

[12] ESP Holding a.s., *Princip radiofrekvenční komunikace RFID*, [online]. [cit. 2016-11-05]. Dostupné z: http://esp.cz/cs/identifikacni-technologie/princip-radiofrekvencni-komunikace-rfid.

[13] Atlas RFID Store, *RFID vs. NFC. What's the difference?*, [online]. [cit. 2016-11-10]. Dostupné z: http://blog.atlasrfidstore.com/rfid-vs-nfc.

[14] Texas Instruments Incorporated, *Near Field Communication*, rok 2016, [online]. [cit. 2016-11-10]. Dostupné z: http://www.ti.com/nfc.

[15] SOS electronic s.r.o., *RDIF čtečka ACM08M(N)*, [online]. [cit. 2016-11-10]. Dostupné z: https://www.soselectronic.cz/products/acm08m-n.

[16] SOS electronic s.r.o., *Klíčenka RFID Unique 125kHz ACM-ABS003BU*, [online]. [cit. 2016-11-10]. Dostupné z: https://www.soselectronic.cz/products/acm-abs003bu.

[17] Raspberry Pi Foundation, *Raspberry Pi 3 Model B*, [online]. [cit. 2016-11-10]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[18] Raspberry Pi Starter Kits, *Raspberry Pi 3 vs Pi 2 vs Pi B+ (Benchmark & Review)*, [online] Dostupné z: https://www.raspberrypistarterkits.com/2016/03/04/raspberry-pi-3-pi-2-pi-b-benchmark-review/.

[19] MUO - Make Use Of, *Operating Systems You Can Run With Raspberry Pi*, [online]. [cit. 2016-11-11]. Dostupné z: http://www.makeuseof.com/tag/7-operating-systems-you-can-run-with-raspberry-pi/.

[20] Lifehacker, *The Best Operating Systems for Your Raspberry Pi Projects*, [online]. [cit. 2016-11-12]. Dostupné z: http://lifehacker.com/the-best-operating-systems-for-your-raspberry-pi-projec-1774669829.

[21] Raspbian community, *Raspbian OS*, [online]. [cit. 2016-11-12]. Dostupné z: https://www.raspbian.org/.

[22] Raspberry Pi Foundation, *INSTALLING OPERATING SYSTEM IMAGES*, [online]. [cit. 2017-04-07]. Dostupné z: https://www.raspberrypi.org/documentation/installation/installing-images/README.md.

[23] Sourceforge.net, *Software ke stažení Win32DiskImager*, [online]. [cit. 2017-04-07]. Dostupné z: https://sourceforge.net/projects/win32diskimager/files/latest/download.

[24] Raspberry Pi Foundation, *Software ke stažení Raspbian OS*, [online]. [cit. 2017-04-07]. Dostupné z: https://www.raspberrypi.org/downloads/raspbian/.

[25] LEHKÝ P., *Technologie tvorby webových aplikací*, Masarykova Univerzita, Fakulta Informatiky, Brno, rok 2007, počet stran 68, [diplomová práce]. [cit. 2016-11-15]. Dostupné z: https://is.muni.cz/th/4246/fi_m/diplomova_prace-technologie_tvorby_webovych_aplikci.pdf.

[26] Jak psát web, *HTML*, [online]. [cit. 2016-11-15]. Dostupné z: https://www.jakpsatweb.cz/html/struktura.html.

[27] World Wide Web Consortium (W3C), *HTML Tags*, [online]. [cit. 2016-11-17]. Dostupné z: https://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html.

[28] KOSEK J., *Historie a vývoj HTML*, [online]. [cit. 2016-11-25]. Dostupné z: http://htmlguru.cz/uvod-historie.html.

[29] DunLog., *Python a WSGI*, [online]. [cit. 2016-11-25]. Dostupné z: https://blog.milde.cz/post/245-python-a-wsgi/.

[30] Jak psát web, *CSS*, [online]. [cit. 2016-11-25]. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html.

[31] IT Network, *Úvod do JavaScriptu*, [online]. [cit. 2016-11-25]. Dostupné z: http://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk.

[32] ROOT.CZ, *Velký test PHP frameworků*, [online]. [cit. 2016-11-29]. Dostupné z: https://www.root.cz/clanky/velky-test-php-frameworku-2008/.

[33] Open Source, *MIT License*, [online]. [cit. 2016-11-29]. Dostupné z: https://opensource.org/licenses/MIT.

[34] KeyCDN.com, *CSS Preprocessors – Sass vs LESS*, [online]. [cit. 2016-12-01]. Dostupné z: https://www.keycdn.com/blog/sass-vs-less/.

[35] Linuxexpres, *Chromium open-source varianta Google Chrome*, [online]. [cit. 2016-12-01]. Dostupné z: https://www.linuxexpres.cz/software/chromium-open-source-varianta-google-chrome.

[36] Github.com, *Apps built on Electron*, [online]. [cit. 2016-12-05]. Dostupné z: http://electron.atom.io/apps/.

[37] Raspberry Pi Foundation, *Raspberry Pi Touch Display*, [online]. [cit. 2016-12-05]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-touch-display/.

[38] Git-scm.com, *Základy systému Git*, [online]. [cit. 2017-04-17]. Dostupné z: https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git.

[39] Attlasian.com, *Basic Git commands*, [online]. [cit. 2017-04-17]. Dostupné z: https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html.

[40] Github.com, *Projekt: Hodoor*, [online]. [cit. 2017-04-17]. Dostupné z: https://github.com/hodooor.

[41] Adafruit.com, *Configuring I2C*, [online]. [cit. 2017-04-26]. Dostupné z: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c.

[42] Raspberrypi Spy, *Adding a DS3231 Real Time Clock To The Raspberry Pi*, [online]. [cit. 2017-04-26]. Dostupné z: http://www.raspberrypi-spy.co.uk/2015/05/adding-a-ds3231-real-time-clock-to-the-raspberry-pi/.

[43] NodeJS.org, *Installing Node.js via package manager*, [online]. [cit. 2017-04-26]. Dostupné z: https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions.

[44] Github.com, *Technical Differences Between Electron and NW.js*, [online]. [cit. 2017-04-26]. Dostupné z: https://github.com/electron/electron/blob/master/docs/development/atom-shell-vs-node-webkit.md.

[45] Pypa.io, *Virtualenv*, [online]. [cit. 2017-05-02]. Dostupné z: https://virtualenv.pypa.io/en/stable/.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AIDC	Automatic Identification and Data Capture
API	Application Programming Interface - rozhraní pro programování aplikací
ARM	Architektura mikroprocesorů
BOZP	Bezpečnost a ochrana zdraví při práci
CLI	Command Line Interface - příkazový řádek
CSS	Cascading Style Sheets
DIY	Do It Yourself - udělej si sám
EM	Electro Magnetic - elektro magnetické
form factor	Aspekt HW dizajnu, zachování rozměrů a dalších fyzických specifikací komponent
GB	Giga Byte - gigabajt
GPIO	General Purpose Input Output
GUI	Graphical User Interface - grafické uživatelské rozhraní
HDMI	High Definition Multimedia Interface
HTML	Hyper Text Markup Language
HW	Hardware
Hz	Herz - základní jednotka frekvence
IoT	Internet of Things - internet věcí
I2C	Inter-integrated circuit
JS	JavaScript
kHz	Kilo Herz - jednotka frekvence, 1000 Hz
MB	Mega Byte - megabajt
MHz	Mega Herz - jednotka frekvence, $1*10^6$ Hz
MySQL	Structured Query Language

NFC	Near Field Communication
NPM	Node Package Manager - manažer balíčků NodeJS
OS	Operační systém
P2P	Peer-to-peer
PHP	Hypertext Preprocessor
Pip	Python Package Index - manažer balíčků Python
RAM	Random Access Memory - operační paměť
RFID	Radio Frequency Indentification
RPi	Raspberry Pi
SoC	System on Chip
SD	SD karta - Secure Digital karta
SW	Software
UI	User Interface - uživatelské prostředí
Update	Části aplikací - balíčky, nebo nové verze, odstraňující chyby z předešlých verzí
USB	Universal Serial Bus
UX	User Experience - uživatelská zkušenost
WLAN	Wireless Local Area Network
WSGI	Web Server Gateway Interface
WWW	World Wide Web

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD NOSIČE

Kořeňový adresář přiloženého CD nosiče obsahuje tyto složky:

1. Adresář s textovou částí práce.
2. ZIP archiv se zdrojovými kódy front-end rozhraní [1].
3. ZIP archiv se zdrojovými kódy aplikace klientského terminálu [2].

- `Zaverecna-prace-latex` ..... Adresář s textovou částí práce
  - `colors.aux` ..... Soubor s nastavením barev
  - `nastaveni_udaju.tex`
  - `sablona-prace.tex` ..... Hlavní soubor pro sazbu diplomové práce
  - `sablona-prace.pdf` ..... Vygenerovaný PDF soubor práce
  - `thesis.sty` ..... Balíček pro sazbu kvalifikačních prací
  - `blokova-schemata` ..... Obrázky blokových schémat
  - `kapitoly` ..... Zdrojové texty rozdělené na kapitoly
    - `aidc.tex`
    - `basic-analysys.tex`
    - `git-github.tex`
    - `modern-web-tech.tex`
    - `navrh-prototyp.tex`
    - `realizace-electron-app.tex`
    - `realizace-frontend.tex`
    - `realizace-terminal.tex`
    - `rpi.tex`
    - `web-tech.tex`
  - `kody` ..... Použité HTML kódy
  - `obrazky` ..... Použité obrázky
    - `front-end` ..... Složka s obrázkami front-end rozhraní
    - `terminal` ..... Složka s obrázkami sestrojení terminálu
    - `terminal-render` ..... Složka s obrázkami terminálové aplikace
  - `pdf` ..... PDF soubory vkládané do práce
    - `podekovani_SIX_DP.pdf`
    - `student-titulka.pdf`
    - `student-zadani.pdf`
  - `text` ..... Hlavní zdrojové soubory
    - `literatura.tex`
    - `prilohy.tex`
    - `reseni.tex`
    - `uvod.tex`
    - `zaver.tex`
    - `zkratky.tex`
  - `tabulky` ..... Použité tabulky

[1] Verze ze dne 29.04.2017

[2] Verze ze dne 15.05.2017

`Hodoor-frontend-master.zip` .. Soubor se zdrojovými kódy front-end rozhraní

- `.gitignore`
- `casual_tests.py`
- `const_data.py`
- `LICENSE.md`
- `manage.py`
- `package.json`
- `README.md`
- `requirements.txt`
- `attendance` ..... Hlavní složka front end
- `deploy_tools`
- `functional_tests` ..... Testy pro front end
- `screenshots` ..... Složka s náhledy
- `ticker`

`Hodoor-terminal-master.zip` .. Soubor se zdrojovými kódy aplikace terminálu

- `.gitignore`
- `functions.js`
- `index.html`
- `LICENSE.md`
- `main.js`
- `package.json`
- `README.md`
- `settings.json`
- `js` ..... Dodatečné JavaScript funkce
- `img` ..... Obrázky
- `font` ..... Použité písmo
- `css` ..... Soubory kaskádových stylů
- `modules` ..... Soubory JS modulů