VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Viacheslav Kulinich

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM VYŠETŘENÍ PACIENTŮ ZALOŽENÝ NA TECHNIKÁCH MHEALTH
PATIENT ASSESSMENT SYSTEM BASED ON MHEALTH TECHNIQUES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Viacheslav Kulinich**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jiří Mekyska, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Viacheslav Kulinich ***ID:*** 186124
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Systém vyšetření pacientů založený na technikách mHealth

POKYNY PRO VYPRACOVÁNÍ:

V rámci práce bude navržen, implementován a otestován systém, který bude využíván k elektronickému záznamu vyšetření pomocí dotazníků. Dotazníky bude možné vyplňovat na tabletech, popř. chytrých telefonech, založených na systému Android nebo iOS. K mobilní aplikaci bude vytvořena serverová část ve frameworku ngx-admin, ve které bude možné spravovat profily pacientů a sledovat jednotlivá skóre z dotazníků. Dotazníky bude také možné exportovat z mobilní aplikace do formátu PDF nebo XLSX. Veškerá komunikace bude zabezpečena.

DOPORUČENÁ LITERATURA:

[1] SMYTH, Neil. Android Studio Development Essentials - Android 7 Edition: Learn to Develop Android 7 Apps with Android Studio 2.2. 1. CreateSpace Independent Publishing Platform, 2016. ISBN 978-1535425339.

[2] DARWIN, Ian. Java Cookbook: Solutions and Examples for Java Developers. 3. O'Reilly Media, 2014. ISBN 978-1449337049.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Jiří Mekyska, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

UPOZORNĚNÍ:
Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Tato bakalářská práce se zabývá vývojem systému vyšetření pacientů, pomocí dotazníků, využívajících metody mHealth. V prvních dvou kapitolách této práce je shrnuta problematika vyšetřování pacientů, funkce, které musí plnit elektronický dotazník, dále jsou porovnány existující typy mobilních aplikací a je vybrán nejvhodnější typ pro elektronický dotazník. Následující kapitoly obsahují proces tvorby webových aplikací, implementaci klientské a serverové části, jejich komunikaci, použité zabezpečovací prvky a stručný popis grafického rozhraní. Výsledkem je funkční progresivní webová aplikace, která byla úspěšně otestována a zveřejněna na internetu.

KLÍČOVÁ SLOVA

Health 4.0, mHealth, vyšetření pacientů, elektronický dotazník, mobilní aplikace, progresivní webová aplikace, PWA, webová aplikace, MongoDB, UIKit, React, Node.js

ABSTRACT

This bachelor thesis is focused on the possibility of examining patients using an electronic questionnaire based on mHealth techniques. In the first two chapters of this work is included the issue of patient examination, the function that must fulfill the electronic questionnaire, the research of existing types of mobile applications and the description of the selected type. The following chapters contain the process of creating web applications, implementation of client and server part, their communication, used security elements and brief description of graphical interface. The result is a functional progressive web application that has been tested successfully and is published on the Internet.

KEYWORDS

Health 4.0, mHealth, physical examination of patients, electronic questionnaire, mobile application, progressive web application, PWA, web application, MongoDB, UIKit, React, Node.js

KULINICH, Viacheslav. *Systém vyšetření pacientů založený na technikách mHealth*. Brno, 2019, 54 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Jiří Mekyska, Ph.D.

Vysázeno pomocí balíčku `thesis` verze 3.03; `http://latex.feec.vutbr.cz`

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Systém vyšetření pacientů založený na technikách mHealth“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jiřímu Mekyskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Úvod **11**

1 Návrh systému vyšetření pacientů **13**
1.1 Neurologické vyšetření pacientů . . . 13
1.2 Dotazník . . . 13
1.3 Požadované funkce dotazníku, které musí plnit mobilní aplikace . . . 13
1.4 Typy mobilních aplikací . . . 14
1.4.1 Nativní aplikace . . . 14
1.4.2 Hybridní aplikace . . . 15
1.4.3 Webové aplikace . . . 15
1.5 Jednotlivé části systému . . . 15
1.6 Požadované zabezpečení mobilní aplikace . . . 16

2 Progressive Web Application **17**
2.1 Programové řešení . . . 17
2.2 Technologie progresivních webových aplikací . . . 18
2.2.1 Service worker . . . 18
2.2.2 Application Shell . . . 18
2.2.3 Web app manifest . . . 19
2.2.4 Push Notifikace . . . 19
2.2.5 Lighthouse . . . 19
2.3 Úložiště . . . 19

3 Proces vývoje webových aplikací **21**
3.1 Vývojové prostředí . . . 21
3.2 Vývoj webových aplikací . . . 21
3.2.1 Frontend . . . 22
3.2.2 Backend . . . 23

4 Bezpečnost **25**
4.1 Princip šifrování . . . 25
4.2 Hašování . . . 27
4.2.1 MD5 . . . 27
4.2.2 Bcrypt . . . 28

5 Komunikace **29**
5.1 Proces komunikace . . . 30
5.2 DNS . . . 31

5.3 Hosting . . . 32
5.4 Název domény . . . 33

6 Implementace serverové části systému 35
6.1 Dynamický typ webové stránky . . . 35
6.2 Node.js . . . 35
6.2.1 Architektura Node.js . . . 36
6.2.2 Instalace Node.js . . . 36
6.3 Databáze . . . 36
6.3.1 SQL . . . 37
6.3.2 NoSQL . . . 37
6.3.3 MongoDB . . . 38

7 Implementace klientské části systému 40
7.1 Použité technologie klientské části . . . 40
7.1.1 UIKit . . . 40
7.1.2 JavaScriptova knihovna React . . . 40
7.2 Uživatelské rozhraní . . . 41
7.2.1 Rozhraní administrátorské . . . 43
7.2.2 Rozhraní pro doktory . . . 44
7.2.3 Rozhraní pacienta . . . 45
7.3 Režim Offline . . . 46
7.4 Testování . . . 47

8 Závěr 48

Literatura 49

Seznam symbolů, veličin a zkratek 52

Seznam příloh 53

A Obsah přiloženého CD 54

Seznam obrázků

1.1 Komunikačni schéma 16
4.1 Ukázka adresního řádku prohlížeče pro HTTPS 25
4.2 Symetrické šifrování 25
4.3 Asymetrické šifrování 26
5.1 Komunikační schéma pro navázaní spojení mezi klientem a serverem . 30
5.2 Komunikační síťový model TCP/IP 31
5.3 Komunikační model s využitím DNS serveru 32
5.4 Struktura IP paketu 33
7.1 Počítačové uživatelské rozhraní 42
7.2 Přihlašovací okno 42
7.3 Mobilní rozhraní 42
7.4 Založení nového testu 43
7.5 Přidávání pacienta 43
7.6 Seznam doktorů 44
7.7 Výsledek testu 44
7.8 Ukázka exportovaného souboru Excel 45
7.9 Ukázka testu 45
7.10 Výsledky všech testů 45
7.11 Instalace aplikace na plochu mobilu 46
7.12 Přidána ikona dotazníku 46
7.13 Lighthouse 47

Seznam tabulek

2.1 Velikost úložiště Cache 20

Seznam výpisů

3.1 Příklad implementace hlavičky HTML souboru. 22
4.1 Příklad haše MD5 27
4.2 Příklad haše bcrypt 28
6.1 Příklad SQL příkazů pro vytvoření tabulky reprezentující pacienty, jejich přidání a následný výpis 37
6.2 Příklad šablony pro vytvoření uživatelů v MongoDB 39
6.3 MongoDB dokument s uloženým pacientem 39
6.4 Příklad identifikátora dokumentu v MongoDB 39

Úvod

Informační technologie se v moderní době rychle rozvíjejí a usnadňují lidem práci v oblasti vědy, vzdělávání, financí a dokonce i medicíny. V medicíně se toto odvětví nazývá eHealth. Hlavní úkoly jsou:

- elektronická zdravotní dokumentace,
- elektronická identifikace pacienta a zdravotnického pracovníka,
- elektronická zdravotnická informační síť,
- elektronické vzdělávání pracovníků.

Jedním z konceptů zdravotní péče eHealth jsou efektivní diagnostické, léčebné a ošetřovatelské procesy, které jsou známé pod pojmem Health 4.0. Jedná se o digitální medicínu obsahující aplikace, které pacientům poskytují lepší péči. Na rozdíl od starších verzí eHealth, je nová verze kombinací všech předchozích použitých technologií se sběrem dat v reálném čase a zvýšeným využitím umělé inteligence. Shromážděné údaje jsou k dispozici lékařům, kteří mají možnost z dat vyčíst informace, které zrovna potřebují, zpracovávat je a přenášet mezi různými zařízeními. Lékaři a pacienti mají k datům přístup kdykoli a kdekoli.

K digitální medicíně patří také mHealth. Pokrývá všechny aplikace telekomunikačních a multimediálních technologií pro poskytování lékařských informací. Tato technologie se používá k monitorování zdravotního stavu člověka, např. srdečních onemocnění, diabetu, autismu, nespavosti a astmatu, pomocí mobilního zařízení. Obvykle se digitální zdravotní aplikace mHealth používají pro [1]:

- diagnostiku a léčbu,
- popis nemocí a epidemií,
- vzdálený sběr dat a vzdálené pozorování,
- řízení chronických onemocnění.

Jedním ze způsobů elektronické diagnostiky pacientů je vyplnění dotazníků. Zda je pacient nemocný, lze zjistit prostřednictvím odpovědí na otázky, týkající se zdravotního stavu pacienta. Pacienti mají možnost procházet elektronické dotazníky jak na počítači, tak i na mobilu nebo tabletu.

Bakalářská práce si klade tyto cíle:

1. Návrh systému vyšetření pacientů na základě technik mHealth.
2. Průzkum všech existujících technologií pro vývoj elektronického dotazníku.
3. Vývoj mobilní aplikace pomocí nejvhodnější technologie.

V následujících kapitolách budou probrány možné technologie pro vyšetření pacientů a bude vybráno nejvhodnější řešení tak, aby byly splněny všechny požadované funkce. V kapitole 1 je popsána problematika vyšetření pacientů a jsou prozkoumány existující technologie pro vývoj systému vyšetření pacientů, jejich výhody a nevýhody. Kapitola 2 popisuje vybranou technologii, její hlavní funkce a proč

byla zvolena. Další kapitoly 4 a 5 popisují komunikaci mezi aplikací a serverem a také zabezpečovací prvky pro zajištění chráněné komunikace. Proces vývoje klientské a serverové části, použité frameworky a programovací jazyky jsou popsané v kapitolách 7 a 6. V kapitole 7 se také nachází popis grafického rozhraní. V závěru práce bude přehledné shrnutí a vyhodnocení splněných cílů.

1 Návrh systému vyšetření pacientů

1.1 Neurologické vyšetření pacientů

Neurologické vyšetření je soubor činností, které lékař provádí za účelem zjištění neurologických poruch. K nálezu neurologických poruch lékaři využívají různé diagnostické metody a analýzy.

Neurologická vyšetření jsou obecně časově a finančně velmi náročná. V závislosti na typu testu jsou pak výsledky k dispozici buď okamžitě nebo s určitým zpožděním. Neurologická vyšetření posuzují motorické a nemotorické schopnosti pacientů jako např. koordinaci pohybů a rovnováhu nebo schopnost člověka mluvit a rozumět. Dále posuzují psychicky stav a změny nálad nebo chování. Při vyšetření lékař sleduje odchylky od normálu a pomocí jednoduchých testů (např. pomocí neurologického kladívka, oftalmoskopu, štětičky a malé baterky pro vyšetření reakce oči na světlo) zhodnotí pacientovi nervové funkce. Podle výsledků testů poté lékař zjistí, jestli pacient trpí takovými poruchami jako např. Parkinsonova nemoc, Huntingtonova nemoc, skleróza nebo epilepsie [2].

1.2 Dotazník

Jedním z nejrychlejších a nejlevnějších způsobů vyšetření pacientů je dotazník. Je to výzkumná metoda pro sběr velkého množství informací prostřednictvím specifických otázek. Za krátkou dobu lze pomocí dotazníku zjistit velké množství informací. V dotazníku se můžou vyskytnout dva typy otázek. První typ jsou otázky uzavřené, ve kterých má tazatel na výběr vícero konkrétních možností. Druhým typem jsou otázky otevřené, které tazateli poskytují volné místo na svobodné vyjádření se. Dříve existovaly dotazníky prakticky pouze v papírové formě, dnes převládají dotazníky elektronické. Elektronické dotazníky totiž mají více výhod oproti papírovým. Hlavními výhodami elektronického dotazníku jsou rychlejší zpracování a praktičtější uchovávání nasbíraných dat.

1.3 Požadované funkce dotazníku, které musí plnit mobilní aplikace

K vývoji aplikace mobilního elektronického dotazníku je zapotřebí si zvolit vhodnou technologii. Volíme ji tak, aby aplikace po dokončení splňovala všechny požadované funkce:

- Přístup k aplikaci. Přístup musí být zabezpečený. Přístup se bude uskutečňovat pomocí loginu a hesla, které jsou registrované na serveru.
- Průběžné ukládání. Každá odpověď se ukládá do zařízení nezávisle na tom, jestli test bude dokončen. Uživatel si může nedokončený dotazník znovu spustit a pokračovat z poslední uložené otázky.
- Ukládání dat. Uživatel bude mít všechny svoje zodpovězené dotazníky uložené na serveru a bude k nim mít zpětně přístup ze svého registrovaného účtu.
- Více typů registrace. Aplikace bude rozlišovat účty pacientů, účty lékařů, účet správce a bude disponovat funkcí přidání nových uživatelů.
- Správa více dotazníků. Uživatel může mít k dispozici více dotazníků. Nové dotazníky přidává správce ze správcovského účtu.
- Automatické vyhodnocení. Každá odpověď na otázku bude mít svou hodnotu. Tyto hodnoty se na konci dotazníku automaticky sečtou a výsledek se zobrazí na obrazovce.
- Provoz mobilní aplikace v režimu offline.
- Mobilní aplikace musí fungovat na všech operačních systémech.

1.4 Typy mobilních aplikací

Existují tři základní typy mobilních aplikací [6]:

1. Nativní aplikace
2. Hybridní aplikace
3. Webové aplikace

Všechny tyto typy splňují většinu požadavků z předchozí kapitoly 1.3. Každý ale má své výhody a nevýhody, na které je třeba se kriticky podívat a zvolit podle nich ten nejvhodnější pro tento mobilní dotazník.

1.4.1 Nativní aplikace

Nejběžnější a nejčastěji používaný typ mobilních aplikací. Pomocí nativních aplikací je možné plně využít funkce telefonu jako např. GPS, videokamera, Bluetooth a různé typy mobilních senzorů. Nejčastěji jsou aplikace distribuovány prostřednictvím obchodů s aplikacemi PlayMarket pro Android a AppStore pro iOS. Hlavními výhodami nativních aplikací jsou rychlost odezvy a schopnost pracovat offline. Nevýhodou těchto aplikací jsou náklady na vývoj, protože nativní aplikace je napsána pro konkrétní operační systém. Aplikace pro Android jsou napsané pomocí programovacího jazyka Java nebo Kotlin a pro iOS pomocí Objective C nebo Swift.

1.4.2 Hybridní aplikace

Hybridní aplikace se velmi podobají nativním aplikacím, ale jsou navrženy na základě webových aplikací. Rozdíl od nativních aplikací spočívá v tom, že základní funkce aplikace jsou naprogramovány a uloženy na serveru a na zařízení klienta je jen minimum. I tyto aplikace lze stáhnout prostřednictvím obchodu mobilních aplikací (Play Market, AppStore). Hlavní výhodou hybridních aplikací je multiplatformnost tzn. mohou běžet na libovolném operačním systému. Vývoj takových aplikací je levnější a rychlejší. Nevýhodou je rychlost aplikace a to, že aplikace nedokáže pracovat offline.

1.4.3 Webové aplikace

Jedná se o webové stránky, které v mnoha ohledech vypadají a působí jako nativní aplikace. Nemohou však plně využívat některé funkce mobilního telefonu např.kameru nebo bluetooth. Webové aplikace jsou spouštěné pomocí prohlížeče a jsou napsány v jazycích HTML, CSS a JavaScript. Není potřeba je stahovat z obchodů s aplikacemi. Stačí přejít na stránku pomocí odkazu. Hlavní výhodou webové aplikace je multiplatformnost. Nevýhodou je, že aplikace vyžaduje připojení k internetu.

S vývojem technologií ale vznikl nový směr v programování webových aplikací, který se nazývá Progressive Web Application (PWA). Hlavní podstatou je, co nejvíce se přiblížit k nativním aplikacím a integrovat výhody obou typů aplikací tzn. být stále multiplatformní a fungovat bez připojení k internetu.

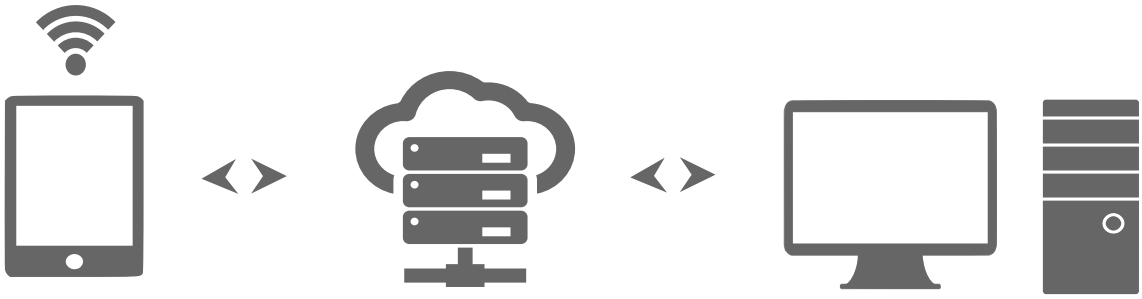
Po prozkoumání všech typů mobilních aplikací bylo zjištěno, že pro vývoj mobilního dotazníku se nejvíce hodí progresivní typ webových aplikaci, který splňuje všechny požadované funkce z kapitoly 1.3.

1.5 Jednotlivé části systému

Systém neurologického vyšetření pacientů založený na technologiích mHealth se bude skládat ze dvou částí:

1. aplikace na mobilním zařízení,
2. serverové části.

Komunikační schéma tohoto systému je zobrazeno na obrázku 1.1. V režimu offline se budou data ze zodpovězených nebo rozpracovaných dotazníků ukládat do paměti mobilního zařízení. Po připojení zařízení k internetu budou data se serverem synchronizována. Data z dotazníků přijata na server se zde budou uchovávat pro další zpracování.



Obr. 1.1: Komunikačni schéma

1.6 Požadované zabezpečení mobilní aplikace

Vzhledem k tomu, že mobilní aplikace shromažďuje údaje o pacientech, je potřeba se zabývat způsobem, kterým mobilní aplikace bude komunikovat se vzdáleným serverem a způsobem, kterým se data na serveru budou uchovávat. Mobilní aplikace a server musí podporovat autentizaci a autorizaci. Přenos dat musí probíhat po zabezpečeném kanálu.

Autorizace je proces poskytnutí jedné osobě nebo skupině osob práva na provedení některé operace nebo povolení přístupu. Procesu autorizace lze dosáhnout pomoci auntentizace. Autentizace je ověření totožnosti uživatele pomocí loginu a hesla (u dvoufázového ověřování navíc ještě dalším kódem). Pro podporu zabezpečeného přenosu datového systému je třeba používat protokol HTTPS. HTTPS protokol je rozšíření protokolu HTTP pro šifrovanou komunikaci. Používá se pro komunikaci mezi web serverem a webovým prohlížečem.

Data uložená na serveru musí být šifrovaná. Šifrování se používá k utajení obsahu dat, ke kterým by mohla získat přístup nedůvěryhodná entita. Nezašifrovaná data jsou šifrovaná pomocí šifrovacích algoritmů a šifrovacích klíčů. Tento proces generuje šifrovaný text. Ten lze zobrazit v jeho původní nezašifrované podobě tak, že se dešifruje pomocí správných klíčů. Dešifrování je jednoduše inverzní šifrování, a to po stejných krocích, ale v obráceném pořadí aplikace klíčů. Proces šifrování se provádí za pomoci symetrických nebo asymetrických šifer [3].

Pokud bude mobilní aplikace zabezpečená těmito prostředky, může být použita pro sběr dat uživatelů.

2 Progressive Web Application

2.1 Programové řešení

Progressive Web Apps (PWA) je technologie, která byla představena Googlem v roce 2015. PWA je jednou z nejpopulárnějších technologií ve Web programování a stále častěji se využívá mezi vývojáři v IT. Díky technologie PWA se internetová stránka může zobrazit uživateli ve formě mobilní aplikace. Uživatel si totiž stránku může stáhnout na mobilní telefon (resp. tablet) a používat ji jako mobilní aplikaci i bez internetového připojení. Pokud je uživatel na pracovní ploše mobilu, používá k prohlížení internetu Chrome a nemá vypnuté vyskakování oznámení, při jeho příchodu na stránku mu sám systém nabídne nainstalování takové aplikace. Tento nový aplikační model kombinuje funkce moderních prohlížečů s výhodami nativních aplikací. Tuto technologii na svých stránkách využívají společnosti jako: Twitter, Trivago, Aliexpress, Uber a Telegram.

Progresivní webové aplikace jsou [7]:

- Spolehlivé (Reliable) – aplikace se načte okamžitě i bez připojení k internetu.
- Rychlé (Fast) – rychlá interakce s uživatelem.
- Sympatické (Engaging) – vypadá jako nativní aplikace, práce s ní je příjemná.

PWA se velice svými charakteristikami podobá nativním aplikacím. Z toho důvodu by práce s takovým dotazníkem měla být příjemnější. Největší výhodou PWA na rozdíl od webových aplikací je, že se dotazník načte rychleji než v prohlížeči. Progresivní aplikace má všechny základní zdrojové soubory uložené do paměti zařízení. Sítí se přenáší pouze měnící se datový obsah. Prohlížeč v tomto případě funguje jako virtuální stroj, který uchovává data a spouští progresivní aplikace, stejně jako Android je virtuálním strojem nativních android aplikací. Funkce, které PWA musí splňovat jsou:

- Progresivní – aplikace běží na každém operačním systému a v každém prohlížeči.
- Adaptivní – aplikace se přizpůsobí každému zařízení (desktop, chytrý telefon, tablet).
- Nezávislé na připojení k internetu – aplikace běží i při nestabilním nebo žádném připojení.
- Musí vypadat jako nativní aplikace – aby uživatelé pracovali s již dobře známým prostředím.
- Samočinně synchronizována – aplikace se musí automaticky synchronizovat se serverem za pomoci Service Worker API.
- Zabezpečená – pomoci protokolu HTTPS se bude předcházet zachycení nechráněných dat.

- Jednoduše instalovatelná – umožňuje instalování aplikace bez používání obchodů s aplikacemi.
- Jednoduchá na používání – abychom spustili aplikaci stačí jen otevřít příslušnou stránku v prohlížeči. Instalace není povinná.

2.2 Technologie progresivních webových aplikací

2.2.1 Service worker

Service worker (SW) je nejdůležitější technologií v Progressive Web Application. Odpovídá za provoz aplikace bez připojení k internetu. Dříve bylo za realizaci offline režimu zodpovědné jiné API – AppCache, které však umožňovalo offline provoz pouze jednostránkovým aplikacím. SW je skript, který pracuje na pozadí prohlížeče nezávisle na uživateli. Tento skript umožňuje sledování síťového provozu stránky, spravování push oznámení[1] a nebo pomocí Cache API běh aplikace offline. Jeho klíčovou vlastností je schopnost zachytit, zpracovat a uložit síťové požadavky do mezipaměti. Právě tato mezipaměť umožňuje aplikaci provoz v offline režimu. Další vlastností SW je, že běží mimo hlavní tok dat JavaScriptu, který řídí aplikaci, a proto jej neblokuje. SW je zcela asynchronní, proto ho není možné používat se synchronními API. Z bezpečnostních důvodů funguje SW pouze přes protokol HTTPS (viz kap. 4).

Service worker je web – worker[2]. Ty se dělí až na tři typy[9]:

1. Dedicated Worker
2. Shared worker
3. Service worker

Web – workery umožňují provádět složité výpočty bez blokování základního pásma uživatelského rozhraní. Tímto způsobem vzniká multithreading[3], který umožňuje provoz, i když je síť odpojena. SW pracuje nezávisle na aplikaci, se kterou je vázán. Může dostávat zprávy, i když aplikace není aktivní. Zprovoznění SW probíhá ve třech krocích: registrace, instalace a aktivace [15].

2.2.2 Application Shell

Application shell je uživatelské rozhraní, které je vytvořeno pomocí HTML, CSS, JavaScriptu. Je důležitou součástí PWA. Application Shell spouští aplikaci v samostatném okně, mimo prohlížeč. Jeho spuštění musí být rychlé a okamžitě kešované.

[1]Push-oznámení jsou krátká oznámení, která se zobrazují na ploše počítače nebo telefonu.

[2]Web – worker je mechanismus, který spouští skripty v podprocesu, tj. v toku na pozadí. To znamená, že tok workeru může provádět úkoly bez zasahování do uživatelského rozhraní.

[3]Multithreading – současné provádění několika úloh jedním programem.

Při prvním spuštění se soubory rozhraní načítají a následně se ukládají do paměti zařízení. Při dalším spuštění se aplikace už načítá z Cache paměti zařízení, což je viditelně rychlejší. App Shell je v podstatě kostrou (šablonou) grafického rozhraní, která by zůstala, kdyby z aplikace odstranili veškerý její obsah a dynamické součásti. [11].

2.2.3 Web app manifest

Web app manifest je soubor JSON, který pro prohlížeč definuje základní informace o aplikaci, jako jsou název, ikona, popis a jiné vlastnosti. Web app manifest je ale také využíván při procesu instalace PWA do uživatelského zařízení. Jeho úkolem je nainstalovat zástupce aplikace na domovskou obrazovku. Tím uživateli poskytne rychlejší přístup k aplikaci. Pro ověření správného nastavení manifestu lze použít panel v Chrome DevTools [12].

2.2.4 Push Notifikace

Notifikace jsou informační zprávy, které vyskakují na obrazovce uživatelského zařízení. Notifikace např. upozorňují uživatele na dostupnost aktualizací. Ke tvorbě push notifikací se používají dvě API: notifikační API a Push API. Notifikační API má za úkol zobrazovat zprávy na obrazovce uživatelského zařízení. Push API umožňuje Service Workeru zpracovávat push zprávy získané ze serveru i při neaktivní aplikaci [14].

2.2.5 Lighthouse

Lighthouse je open – source automatizovaný nástroj sloužící ke zlepšení kvality webových stránek. Nástroj může být spuštěn v prostředí Chrome DevTools a poskytuje nám informace o kontrolované stránce z hlediska výkonnosti, přístupnosti a progresivnosti. Pro testování web aplikace stačí pouze vložit URL stránky do Lighthousu. Výsledkem je zpráva o tom, jak dobře byla aplikace napsána, a co všechno a jak by se dalo ještě zlepšit [10].

2.3 Úložiště

Technologie Service Worker, která je zodpovědná za uchovávání dat v zařízení, je asynchronní. Asynchronní API znamená, že dostupnost zdroje, služeb nebo úložišť nemusí být okamžitá. Existují pouze dva typy asynchronních úložišť: Cache API a IndexedDB.

IndexedDB je asynchronní a objektově orientovaná databáze, která umožňuje jednoduše ukládat a načítat data indexované klíčem.

Cache API je systém pro ukládání a načítání síťových požadavků a odpovědí. Tyto požadavky a odpovědi mohou být obyčejné, vytvořené za účelem běhu aplikace, nebo mohou být vytvořené výhradně za účelem uložení dat v Cache paměti. Když je zapotřebí vymazat všechna uložená data, slouží k tomu vývojářský panel v prohlížeči Google Chrome a položka s názvem „Application". Pro načítání síťových zdrojů aplikace v offline režimu se používá Cache API, která má limitovanou velikost úložiště (viz tab. 2.1) [13].

Tab. 2.1: Velikost úložiště Cache

Prohlížeč	Limit
Chrome	<6% volného prostoru
Firefox	<10% volného prostoru
Safari	<50 MB
IE10	<250 MB

3 Proces vývoje webových aplikací

V této kapitole budou popsány procesy a technologie, které se typicky využívají k vývoji webových aplikací jak na serveru tak u klienta.

3.1 Vývojové prostředí

Vývojové prostředí je speciální program určený pro celý cyklus psaní a testování programů v konkrétním jazyce. Typické vývojové prostředí obsahuje: textový editor, správce souborů, kompilátor. K vývoji webových aplikací postačí pouze textový editor, který upravuje textový obsah souboru. Nejpoužívanější jsou tyto [24]:

- Sublime Text 3
- Atom
- VS Code
- WebStorm

Každý textový editor má své výhody a nevýhody. Hlavní výhody používání textových editorů jsou: zvýraznění syntaxe, automatické odsazení, automatické dokončování a také plugin emmet, který může sbalit velký kus kódu do jednoho řádku, což usnadňuje čtení kódu.

3.2 Vývoj webových aplikací

Pro vývoj mobilních médií byl zvolen progresivní typ aplikace. V podstatě je to webová stránka, která pomocí technologií popsaných v kapitole 1.3 transformuje webovou stránku do mobilní aplikace. Vytvoření mobilní progresivní aplikace je proto založeno na principu vytváření webové stránky.

Veškerý vývoj webových aplikací je rozdělen do dvou částí: frontend (klient) a backend (server) a lze jej provádět několika způsoby:

1. Vytváření webových stránek pomocí konstruktérů. Tato metoda je nejjednodušší a nevyžaduje od vývojáře v programování mnoho znalostí. Princip práce s konstruktéry je takový, že stačí vybrat šablonu a přizpůsobit ji potřebám. Tato metoda však není vždy nejvhodnější, protože možnosti takové webové stránky jsou dosti omezené a stačí jen pro nejjednodušší aplikace. Nejpoužívanější službou je WordPress.
2. Naprogramování webové stránky. Vývoj takové aplikace je složitější, protože potřebujeme základní znalosti programování a moderních technologií používaných pro vývoj webových aplikací. Velkou výhodou však je, že touto metodou lze realizovat prakticky cokoli. Programátor není omezen šablonou.

3.2.1 Frontend

Frontend to je vývoj uživatelského rozhraní. Jedná se o vytvoření webové stránky nebo webové aplikace s použitím HTML, CSS a JavaScriptu tak, aby uživatel mohl s webovou stránkou přímo komunikovat a pracovat. Každý web je tvořen určitými vrstvami struktur, dat, designu, funkčnosti a obsahů. Kombinováním značkovacích jazyků, designu, klientských scénářů a frameworků vývojář vytváří prostředí pro uživatele tak, aby bylo vše přehledné a funkční [4]. Pro psaní frontendu je zapotřebí: HTML, CSS a JavaScript.

HTML

HTML (HyperText Markup Language) je standardní značkovací jazyk pro tvorbu WWW stránek. Pomocí tohoto jazyka se tvoří základní struktura stránky: nadpisy, odstavce, seznamy a podobně. HTML kód je kombinací normálního textu a značek. Smysl HTML značek je v oddělování a vyznačování částí textu. Značkám se také říká tagy. Jazyk HTML je interpretován prohlížečem a zobrazuje se na monitoru počítače nebo mobilního zařízení jako dokument. Stránky HTML jsou přenášeny prohlížeči ze serveru pomocí protokolů HTTP nebo HTTPS. V případě HTTP jde o přenos prostého textu, v případě HTTPS pak jde o text zašifrovaný. Dokumenty napsané v jazyce HTML mají příponu .html nebo .htm. Příklad hlavičky standardního HTML souboru je zobrazen ve výpisu 3.1.

Výpis 3.1: Příklad implementace hlavičky HTML souboru.

```
<!DOCTYPE html>                                                                    1
<html lang="en">                                                                   2
    <head>                                                                         3
    <meta charset="utf-8">                                                         4
    <title>Dotazník</title>                                                        5
    <link href="css/uikit.min.css" rel="stylesheet">                               6
    <link href="css/style.css" rel="stylesheet"                                    7
    </head>                                                                        8
    <body>                                                                         9
        <header>                                                                   10
        </header>                                                                  11
        <main>                                                                     12
        </main>                                                                    13
        <script type="text/javascript" src="libs/uikit.min.js"></script>           14
        <script type="text/javascript" src="js/common.js"></script>                15
    </body>                                                                        16
</html>                                                                            17
```

Prvek <! DOCTYPE> je určen k označení typu dokumentu a je standardní. Tag <html> definuje začátek souboru HTML, uvnitř kterého jsou uloženy hlavička <head> a tělo dokumentu <body>. Tag <head> obsahuje povinné prvky, které definují vlastnosti stránky:

- <meta charset = „utf-8“> označuje kódování stránky a je standardní.

- <title> definuje název webové stránky, který se zobrazí na kartě prohlížeče.
- <link ...> zapojuje grafické soubory jako: UIKit a CSS.

Tělo dokumentu <body> je určeno pro umístění tagů a obsahové části webové stránky. <header> je navigační panel webu a element <main> obsahuje obsah stránky. <script> zapojuje knihovny, frameworky typu UIKit a soubor JavaScript, který je zodpovědný za funkci webu [26].

CSS

CSS (Cascading Style Sheets) je formální jazyk, který popisuje vzhled dokumentu napsaného pomocí značkovacích jazyků jako jsou HTML, XHTML nebo XML. CSS se při tvorbě webových stránek používá k určení barev, písem a umístění jednotlivých bloků a následně k zobrazení vzhledu těchto stránek. CSS umožňuje prezentovat stejný dokument různými styly. Při práci s dokumenty HTML používá CSS principy dědičnosti a kaskádování. Princip dědičnosti znamená, že všechny následující prvky dokumentu dědí stylové parametry prvku předchozího. Kaskádovacím principem se rozumí to, že jednomu prvku HTML může být přiřazeno více než jedno pravidlo CSS, a ty se kupí na sebe v tzv. kaskádě.

JavaScript

JavaScript je skriptovací programovací jazyk. Podporuje objektově orientovaný typ programování. Typicky se tento programovací jazyk používá pro přístup programu k objektům aplikace. V prohlížečích je velmi rozšířený jako skriptovací jazyk pro interaktivní webové stránky. Jeho úkolem je reagovat na akce uživatele, zpracovávat kliknutí myší, posunutí kurzoru nebo stisknutí kláves. Odesílá také požadavky na server a načítá data bez opětovné aktualizace stránky. Struktura jazyka sestává ze tří části: jádro, objektový model prohlížeče a objektový model dokumentu.

Pro zjednodušení psaní kódu se používají podpůrné technologie, jako jsou frameworky a knihovny. Knihovna může být použita v softwarovém produktu jednoduše jako soubor podsystémů s podobnou funkčností, aniž by to ovlivnilo architekturu hlavního softwarového produktu, zatímco framework diktuje ve tvorbě architektury aplikace určité pravidla. Nejpoužívanější frameworky pro tvorbu uživatelských rozhraní jsou: AngularJS, Bootstrap, UIKit, Foundation, Skeleton. Nejpoužívanější knihovny jsou: React, jQuery a Gulp.

3.2.2 Backend

Backend je soubor hardwarových a softwarových nástrojů, v nichž je implementována logika práce webu. Tady se realizuje práce s datovým úložištěm a vývoj logiky

serveru. Na serveru závisí výkon a zabezpečení webové aplikace. Pro psaní logiky webu se používají serverové programovací jazyky. Nejpopulárnější jsou: PHP, Ruby, Python a Java. Pro zjednodušení vývoje serverů se používají frameworky stejně jako i ve frontendu. Nejpopulárnější jsou: Yii, Node.js, Ruby on Rails a .NET Framework. Každý jednotlivý framework používá určitý programovací jazyk [35].

Nejpoužívanějším řešením pro ukládání dat z webové aplikace jsou databáze. Databáze (DB) je organizovaná struktura pro ukládání, modifikaci a zpracování informací. DB se používají pro dynamické weby, které shromažďují data. K vývoji backendu se používají různé systémy pro správu databází: MySQL, PostgresSQL, SQLite a MongoDB.

Nejdůležitějším prvkem vývoje backendu je webový server. Jedná se o speciální program, který přijímá příchozí http – požadavky, spustí kód a vrátí generovanou odpověď. Klient, který odesílá požadavky, je obvykle webový prohlížeč. Odešle na webový server požadavky na zdroje označené URL adresami. Zdroje jsou stránky HTML, archivy, audio, video a soubory. Výměna probíhá prostřednictvím protokolu HTTP nebo HTTPS. Funkce, které může webový server provádět:

- automatizace webových stránek,
- vedení záznamů uživatelského přístupu ke zdrojům,
- autentizace a autorizace uživatelů,
- podpora pro dynamicky generované stránky,
- podpora protokolu HTTPS pro zabezpečená připojení klienta.

4 Bezpečnost

Důležitou podmínkou pro fungování progresivní webové aplikace je podpora protokolu HTTPS. Protokol HTTPS poskytuje bezpečnost a integritu dat pro webové stránky a osobní informace uživatelů. HTTPS pomáhá bránit útočníkům v získání informací putující mezi prohlížečem a serverem. Útočníci využívají nezabezpečenou komunikaci, aby získali z uživatele jeho soukromé údaje nebo nainstalovali škodlivý software se stejným účelem.

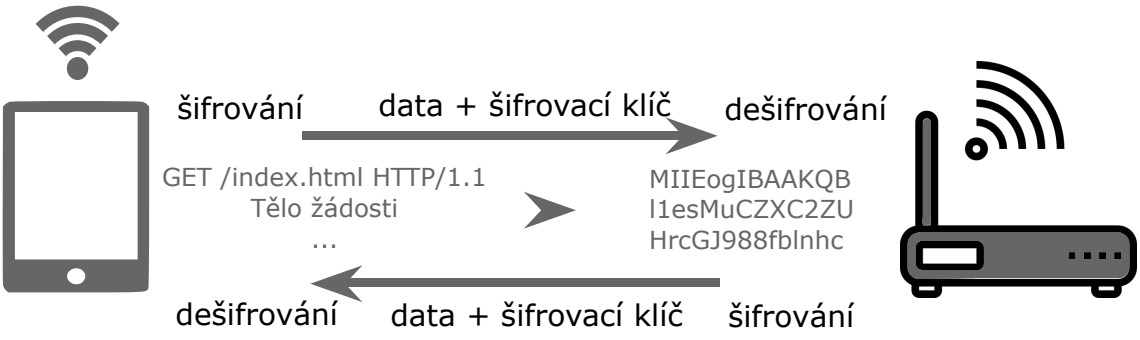
HTTPS využívá protokol HTTP spolu s protokolem SSL nebo TLS a jako standard se zde bere certifikát X.509. Protokol SSL zajišťuje identifikaci. Díky SSL jsou si klient a server jisti, že komunikují opravdu mezi sebou a ne s útočníkem, který se za ně vydává. Protokol také zajišťuje šifrovanou komunikaci. Poskytuje tedy šifrovacímu algoritmu prostředky, které klient a server využívají k bezpečné výměně klíčů [16]. Jaký komunikační protokol daná webová aplikace využívá, lze jednoduše zjistit z adresního řádku prohlížeče (obr. 4.1).

Obr. 4.1: Ukázka adresního řádku prohlížeče pro HTTPS

4.1 Princip šifrování

Pro šifrování dat jsou zapotřebí [17]:

1. Data k šifrování
2. Unikátní šifrovací klíč
3. Šifrovací algoritmus

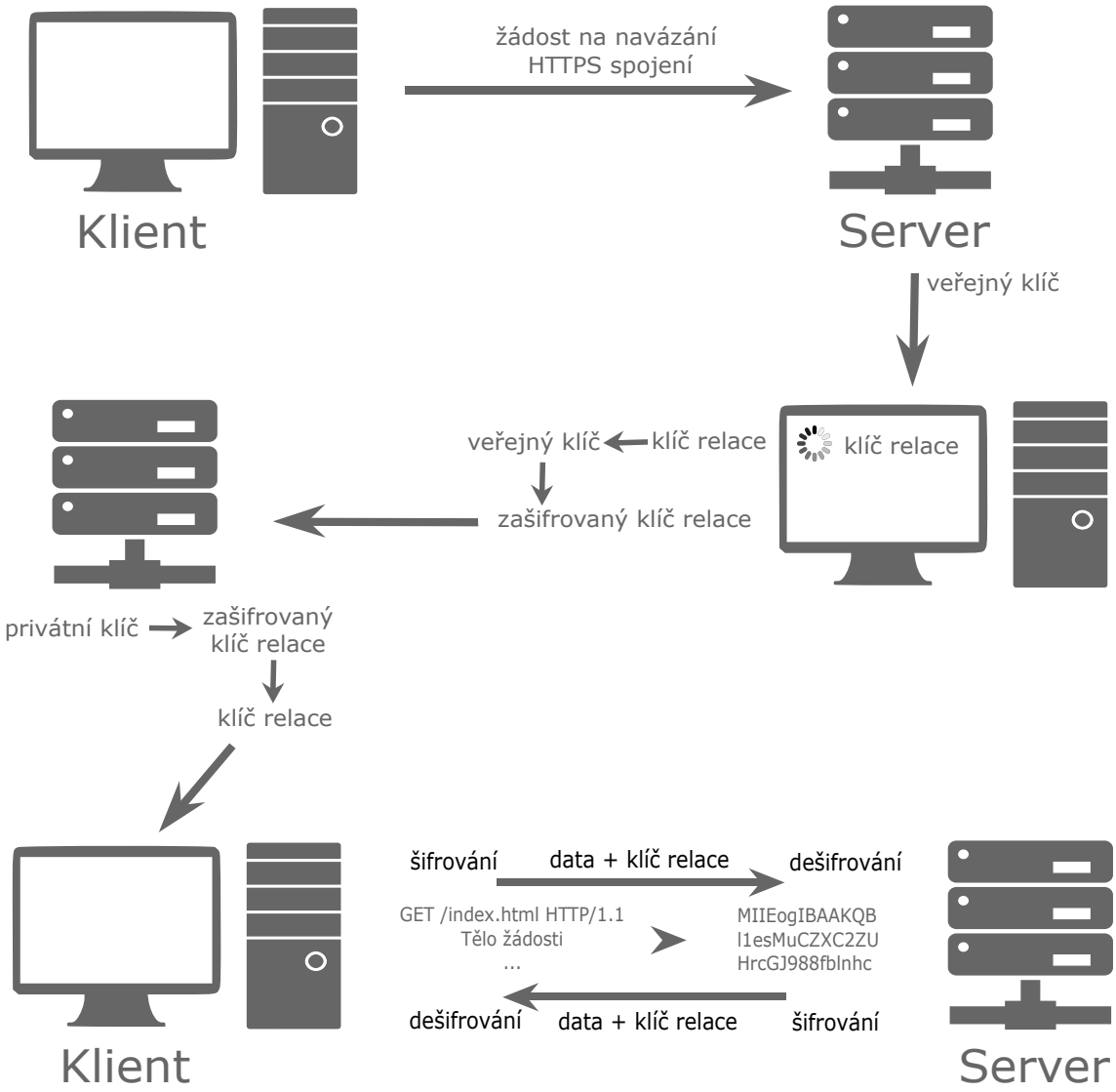


Obr. 4.2: Symetrické šifrování

Do šifrovacího algoritmu se vkládají data a šifrovací klíč. Výstupem je nečitelný šifrovaný text. Pro dešifrování šifrovaného textu stačí provést obrácený proces pomocí

stejného klíče. Když se stejný klíč používá na obou koncích, jde o symetrické šifrování. Tímto způsobem funguje např. domácí WIFI síť (obr. 4.2). V případě webové aplikace ale tento způsob šifrování není ideální, jelikož není možné kontrolovat druhý konec připojení. Nedá se zjistit, jestli útočník šifrovací klíč někde uprostřed komunikace nezachytil. Proto je v tomto případě lepší používat asymetrické šifrování.

Asymetrické šifrování využívá dva různé klíče, první pro šifrování a druhý pro dešifrování. Jelikož se tímto způsobem komunikace šifruje na veřejném internetu, je označována jako „Public Key Cryptography". Použitím asymetrického šifrování se eliminuje potřeba výměny šifrovacího klíče mezi klientem a serverem. Nevýhodou asymetrického šifrování je vyšší výpočetní náročnost. Proto se běžně tyto dva typy šifrování kombinují a využívají současně.



Obr. 4.3: Asymetrické šifrování

Schéma asymetrického šifrování je zobrazeno na obrazku 4.3 a postup je násle-

dující [17]:

1. Prohlížeč klienta požádá o připojení na webový server.
2. Server odešle klientovi svůj veřejný klíč, zatímco svůj soukromý klíč udržuje v tajnosti.
3. Prohlížeč vygeneruje třetí klíč, tzv. klíč relace (Session Key).
4. Tento klíč relace je potom zašifrován počítačem pomocí veřejného klíče získaného ze serveru.
5. Následně je zašifrovaný klíč relace sdílen se serverem.
6. Server dešifruje klíč relace pomocí tajného soukromého klíče.
7. Dešifrování klíče relace je ukončeno a nahrazeno symetrickým šifrováním.

Nejpoužívanější technikou asymetrického šifrování je RSA. Protože je ale soukromý klíč RSA 2048 bitový, hodně zpomaluje proces šifrování. Proto se častěji používá symetrický klíč AES, který má pouze 256 bitů.

4.2 Hašování

Dalším zabezpečením webové aplikace je autentizace. Po zobrazení webové stránky prohlížeč uživatele žádá o přihlášení. Toto přihlášení je nutné v případě, že chceme v aplikaci pracovat. Typickými vstupními údaji jsou login a heslo (viz obr. 7.2). Po vyplnění těchto polí začne aplikace porovnávat zadané údaje s údaji v databázi. V případě shody dat aplikace uživateli umožní přístup.

Ke zvýšení bezpečnosti procesu autentizace se dále využívá šifrování uživatelova hesla v databázi. Provádí se tzv. hašováním. Hašování je matematický algoritmus, který převádí libovolné pole dat na řetězec pevné délky skládající se z písmen a čísel. Kromě toho zůstává při použití stejného typu hašování tato délka nezměněna bez ohledu na množství vstupních dat. Nejčastěji používaný typy hašování jsou MD5 a SHA.

4.2.1 MD5

MD5 je 128-bitový hašovací algoritmus vyvinutý v roce 1991 (viz výpis 4.1). Vlastnosti hašování MD5:

- MD5 haš obsahuje 32 znaků (písmena anglické abecedy a čísla).
- MD5 je unikátní.
- Hašování MD5 je nevratné.
- Haše jsou statické. Výsledkem hašování stejných dat je vždy stejný haš.

Výpis 4.1: Příklad haše MD5

```
955DB0B81EF1989B4A4DFEAE8061A9A6
```

Výhodou hašování je, že pro útočníka je skutečné heslo obtížně zjistitelné, jelikož MD5 algoritmus je nevratný proces. Inverzní transformaci lze provést pouze triviálním způsobem, a to náhodným zkoušením kombinací. Je zapotřebí spustit cyklus, který bude hašovat všechny možné hodnoty a bude porovnávat výsledek s existujícím hašem. Útočník např. bude muset otestovat všechny možné kombinace, které jsou složené z 8 velkých a malých písmen. Na prvním místě může být 26*2 = 52 znaků. Ke zjištění hledané osmimístné kombinace bude zapotřebí otestovat 52^8 variant. V případě kombinování písmen i s čísly bude tato hodnota ještě vyšší. Proto se doporučuje vytvářet hesla, které jsou sestavené z velkého množství znaků různých typů v různé posloupnosti [18].

V dnešní době už to ale není nejlepší řešení šifrování hesel. Vznikly totiž tzv. duhové tabulky (rainbow table), které obsahují slovníky s hesly a jejich odpovídajícími zahašovanými hodnotami. Tyto tabulky lze snadno stáhnout na veřejném internetu [19].

Nevýhodu MD5 řeší modernější hašovací funkce bcrypt. Tato funkce byla nakonec vyhodnocena jako nejvhodnější a zvolena pro použití v elektronickém dotazníku.

4.2.2 Bcrypt

Bcrypt byla prezentována v roce 1999. Je založena na šifře Blowfish a obsahuje tzv. kryptografickou sůl, která chrání haše proti úspěšnému použití duhových tabulek a zabraňuje kolizím stejných hesel. Blowfish je symetrická bloková šifra, byla prezentována v roce 1993 a využívá proměnnou délku klíče. Sůl je datový řetězec, který se spolu s heslem účastní šifrování. U bcrypt se jedná o 128-bitovou sůl.

Pro šifrování hesla se používá vylepšený Blowfish, který se nazývá eksblowfish. Fáze šifrování sice probíhá déle než u standardní šifry Blowfish, ale to samé platí i pro dešifrování. Útočníkům tedy bude trvat dešifrování více času. V případě krátkého uživatelského hesla ho hašovací funkce rozšíří. Příklad haše statického MD5 je zobrazený ve výpisu 4.1. Příklad měnícího se haše bcrypt je ve výpisu 4.2, kde „$2y$“ je prefix bcryptu, 10 je šifrovací úroveň, „JOGlvZilcTGWJFWECPQSnO“ je sůl a zbytek je hodnota šifrovaná pomocí funkce base64[1] [20].

Výpis 4.2: Příklad haše bcrypt

```
$2y$10$JOGlvZilcTGWJFWECPQSnOckOsPalQN8nGvIwqyv.tn3MqeXbHgO
```

[1]Base64 je kódování, které převádí binární data pouze do 64 znaku ASCII.

5 Komunikace

Komunikace mezi webem a serverem se provádí pomocí protokolu HTTP nebo zabezpečeného protokolu HTTPS.

HTTP (HyperText Transfer Protocol) je protokol přenosu hypertextových dokumentů, který pracuje na aplikační vrstvě. Je to protokol typu klient-server. To znamená, že existuje klient, který iniciuje připojení a pošle požadavek na server, který čeká na spojení. Když požadavek dostane, provede potřebné operace a vrátí klientovi zprávu s výsledkem. Na serveru se používá port TCP/80. V moderní době se používá verze HTTP/2. Nejdůležitějším objektem přenosu protokolu HTTP je zdroj, na který odkazuje URL v žádosti klienta. Zdrojem obvykle jsou soubory uložené na serveru. Zvláštností protokolu HTTP je možnost zadat v žádosti a odpovědi způsob reprezentace stejného zdroje různými parametry: formát, kódování a jazyk. Výměna zpráv probíhá podle schématu požadavek – odpověď [21, 167].

Dotazovací metody protokolu HTTP představují způsoby, kterými se specifikují požadavky klienta na webový server. Obvykle se jedná o krátké anglické slovo. Server může používat libovolné metody. Pokud server nerozpozná metodu zadanou klientem, vrátí kód chyby „501“. Základní metody jsou:

- OPTIONS – používá se pro dotaz na možnosti webového serveru nebo parametry připojení pro konkrétní zdroj. Ke zjištění možností celého serveru, musí klient zadat hvězdičku „*“ do adresy URL. Výsledky této metody nejsou uloženy v mezipaměti.
- GET – slouží k dotazování na obsah z požadovaného zdroje. Touto metodou lze také spustit libovolný proces. V tomto případě by měl být průběh procesu zahrnut do těla odpovědi.
- HEAD – používá se stejným způsobem jako metoda GET s tou výjimkou, že v odpovědi serveru není žádné tělo. Tato metoda se používá k extrahování metadat.
- POST – slouží k přenosu uživatelských dat do zadaného zdroje.
- PUT – slouží k načtení obsahu požadavku na adresu URL uvedenou v žádosti. Pokud prostředek na zadané adrese URL neexistuje, vrátí server stav „201“.
- DELETE – odstraní zadaný zdroj.
- TRACE – vrací přijatou žádost tak, aby se klient mohl přesvědčit, jaké informace k požadavku přidávají nebo upravují servery ležící mezi klientem a zdrojovým serverem.

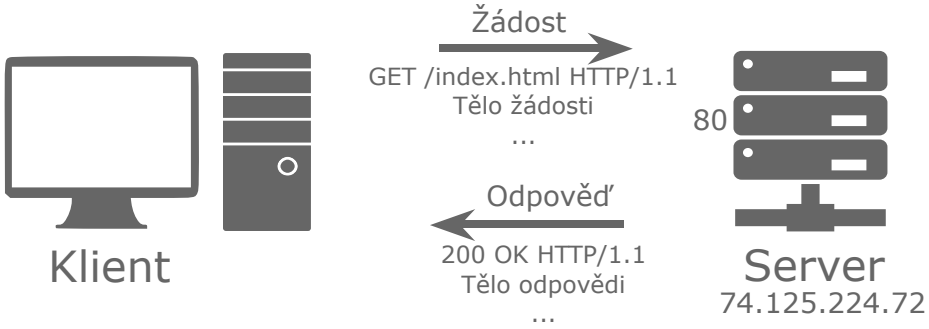
Existují také stavové kódy, které jsou součástí prvního řádku odpovědi serveru. Je to celé číslo složené ze tří číslic. První označuje třídu stavu. Na základě třídy stavu klient zjistí stav odpovědi na svou žádost a v případě potřeby rozhodne, jaké kroky podnikne.V současné době existuje pět hlavních tříd kódů stavu:

- 1xx průběh,
- 2xx úspěch,
- 3xx přesměrování,
- 4xx chyba klienta,
- 5xx chyba serveru.

5.1 Proces komunikace

Navázání spojení mezi prohlížečem a webovým serverem probíhá v následujících krocích (obr. 5.1):

1. Před zahájením relace mezi web klientem a web serverem se nejdříve prohlížeč obrátí na DNS server, ze kterého zjistí, na jaké IP adrese se server nachází.
2. Prohlížeč do získané IP adresy přidává číslo portu. U HTTP protokolu se používá port 80. Konečná adresa může mít tvar 74.125.224.72:80.
3. Prohlížeč navazuje pomocí zjištěné adresy TCP spojení se serverem.
4. Prohlížeč odesílá HTTP GET žádost na server.
5. Server generuje odpověď a posílá ji zpět prohlížeči.
6. Po získání potřebných dat prohlížeč ukončí relaci.

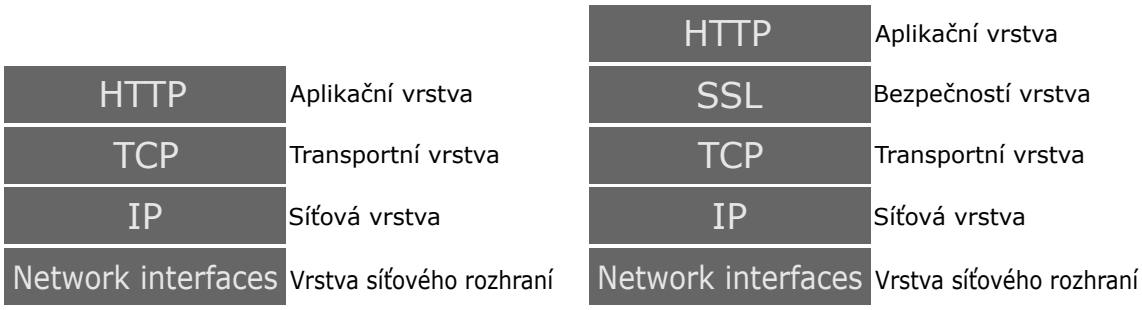


Obr. 5.1: Komunikační schéma pro navázaní spojení mezi klientem a serverem

HTTP relace využívá TCP protokol transportní vrstvy. TCP protokol je spolehlivý transportní protokol, ktery zaručuje, že všechna vyslaná data budou doručena příjemci. Využívá mechanismus potvrzování přijatých dat a opětovných přenosů. Před samotným přenosem dat je navázáno spojení s protější stranou a po dokončení přenosu je spojení rozvázáno. TCP také poskytuje HTTP spolehlivý bitový kanál. Bajty vysílané z jedné strany jsou přijímány druhou ve správném pořadí a beze ztrát.

TCP odesílá data v malých blocích, které se nazývají IP pakety nebo IP datagramy. HTTP protokol je horní vrstvou síťového modelu TCP/IP. Zabezpečená

varianta HTTPS využívá navíc šifrovací vrstvu (TLS nebo SSL), která se nachází mezi HTTP a TCP (Obr. 5.2).



Obr. 5.2: Komunikační síťový model TCP/IP

Když chce HTTP přenést zprávu, odesílá postupně data přes navázané TCP spojení. TCP rozděluje datový proud do bloků, kterým se říká segmenty, a transportuje je uvnitř IP paketů.

Každý segment TCP se přenáší pomocí IP paketů ze zdrojové IP adresy na cílovou IP adresu. Každý IP paket obsahuje:

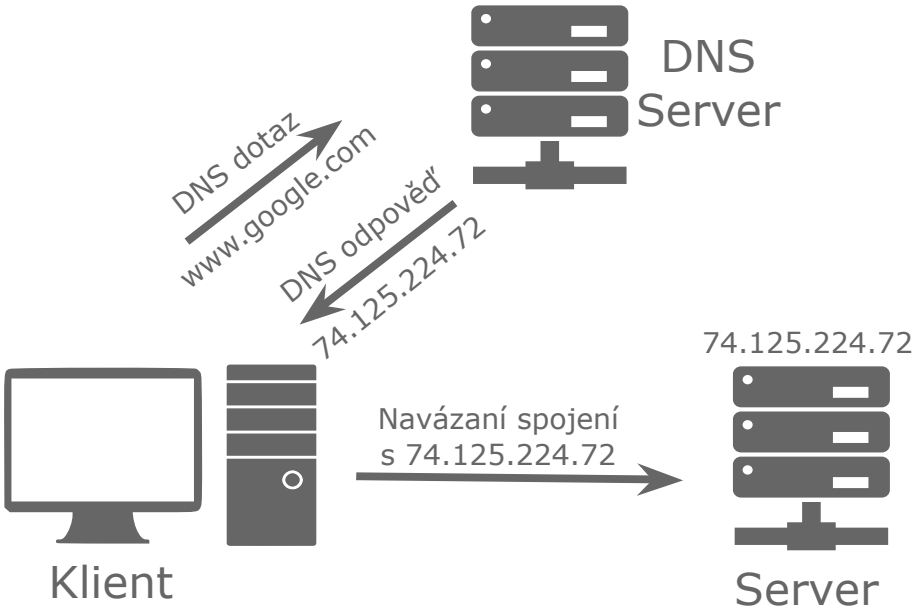
- záhlaví paketu (obvykle 20 bajtů),
- záhlaví segmentu (obvykle 20 bajtů),
- data TCP (0 a více bajtů).

Záhlaví IP paketu obsahuje zdrojové a cílové adresy, informace o velikosti a další příznaky. Záhlaví segmentu TCP obsahuje čísla portů TCP, kontrolní příznaky TCP a číselné hodnoty pro správné seřazení segmentů (viz obr. 5.4) [22, 74].

5.2 DNS

DNS (Domain Name System) je systém doménových jmen, který operuje v aplikační vrstvě. Využívá transportní porty UDP/53 a TCP/53. Jeho úkolem je překlad jmen serverů na jejich IP adresy a naopak pomocí vazeb mezi jmény a IP adresami. Systém využívá princip klient-server. DNS serverů existuje mnoho a jsou organizovány hierarchicky. Bez DNS by uživatel při každém zobrazení jakékoliv webové stránky musel zadávat do prohlížeče IP adresu serveru, na kterém je obsah stránky uložen. Další výhodou je, že v případě změny IP adresy serveru nemusí uživateli vzniknout problém s hledáním nové IP adresy. Aktualizovaný DNS server se totiž sám změně přizpůsobí, ani by toto uživatel postřehl. V případě, že se uživatel pokouší kontaktovat nějakou webovou stránku (např. www.google.com), vyšle prohlížeč na DNS server žádost pro zjištění IP adresy. Ten prohledá svou databázi a v případě, že nenalezne potřebný záznam, začne komunikovat s jinými servery. Po získání IP adresy hledaného serveru odešle odpověď s danou adresou klientovi. Komunikace klienta

a DNS serveru a také komunikace mezi servery probíhá přes transportní protokol UDP. Zjednodušená komunikace je zobrazená na obrázku 5.3 [21, 157].



Obr. 5.3: Komunikační model s využitím DNS serveru

5.3 Hosting

Aby byl web neustále aktivní a pracoval v režimu přerušení, musí být umístěn na hosting. Hosting je služba poskytující prostředky pro umístění informací na server, který je neustále připojený k síti. Jedná se o server, který je neustále spuštěn, má statickou IP adresu, a na kterém je spuštěn software, který je vyžadován pro zpracování požadavků na uložené soubory. Hlavní typy hostingu [23]:

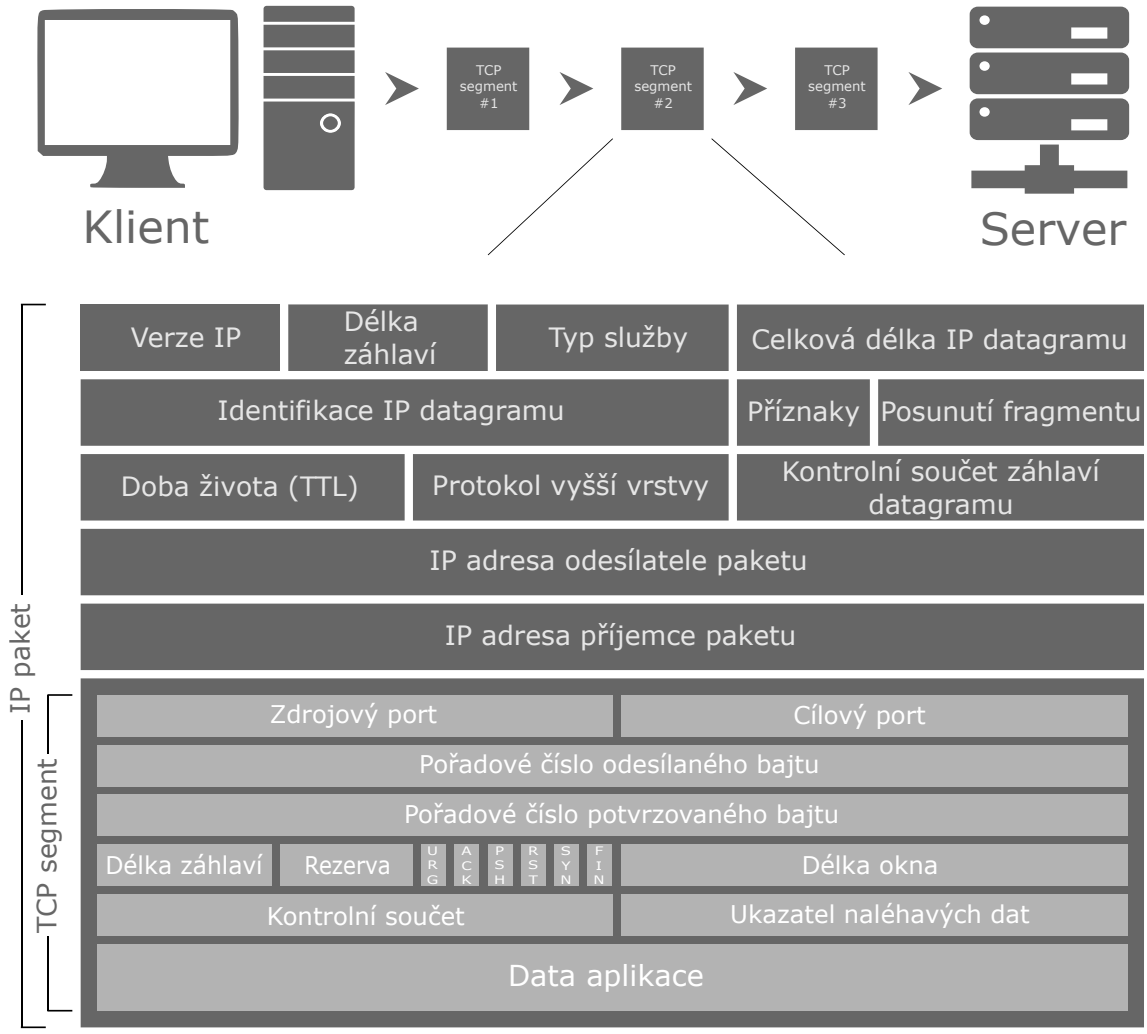
- Hosting souborů. Služba, ve které poskytovatel poskytuje místo pro ukládání informací ve formě souborů, zajišťuje jejich zabezpečení a vzdálený přístup.
- Hosting e-mailu. Služba zajišťující soukromý poštovní systém, který je chráněn proti spamu a reklamě. Pracuje na nejběžnějších protokolech přenosu pošty jako jsou: SMTP, IMAP a POP3.
- Hosting DNS. Služba, která poskytuje úložiště pro záznamy o doméně.
- Herní hosting. Poskytování různých typů serverů na provoz online her pro více hráčů.
- Web hosting. Služba, která jednotlivcům a organizacím umožňuje vytvářet a provozovat vlastní webové stránky na internetu.

Existují placené i bezplatné hostingy. Liší se svými parametry:

- velikost disku,

- limit počtu měsíčních návštěv,
- počet stránek, které mohou být umístěny pod jedním účtem,
- počet FTP uživatelů,
- počet e-mailových schránek a volného prostoru na disku,
- počet současných procesů na uživatele.

Webová aplikace byla zveřejněna na hostingu `https://www.netlify.com/`, který už má nainstalovaný certifikát SSL.



Obr. 5.4: Struktura IP paketu

5.4 Název domény

Aby mohl web fungovat a uživatelé ho mohli najít, je nutné, aby webové stránce bylo přiřazeno individuální jméno - doména. Doména je symbolické jméno, které se

používá k identifikaci oblastí na internetu. Společný jmenný prostor Internetu funguje prostřednictvím služby DNS (viz kap. 5.2). Domény jsou rozděleny do úrovní, které jsou odděleny tečkami. Například feec.vutbr.cz [21, 167]:

- .cz - doména první (horní) úrovně,
- vutbr.cz - doména druhé úrovně,
- feec.vutbr.cz - doména třetí úrovně.

Pro webovou aplikaci byla použita doména „mydotaznik.tk“, pomocí které lze aplikaci snadno najít na internetu. Doména byla bezplatně zaregistrovaná na stránce `https://freenom.com/`.

6 Implementace serverové části systému

V této kapitole je popsána implementace serverová část (Backend) progresivní webové aplikace. Jak již bylo psáno v kapitole 1.5 standardní webová aplikace se skládá ze dvou základních částí: klientské a serverové. Klientská část, jejíž implementace je popsána v kapitole 7, realizuje uživatelské rozhraní. Serverová část aplikace implementuje logiku chování aplikace. Komunikace mezi jednotlivými částmi probíhá skrz žádosti a odpovědi (HTTP protokol viz kap. 5). Jestliže klient potřebuje jakoukoliv informaci, posílá žádost serverové části. Ta žádost přijme, zpracuje, vygeneruje odpověď a pošle ji zpátky klientovi. Základní funkce serverové části jsou:

- příjem žádostí od klienta,
- zpracovávání žádostí a odesílání odpovědí klientovi,
- kontrola přístupu na základě loginu a hesla (autentizace a autorizace),
- podpora dynamicky generovaných stránek,
- práce s databází,
- podpora zabezpečené komunikace HTTPS.

6.1 Dynamický typ webové stránky

Rozlišují se dva typy webových stránek: statické a dynamické. Statické jsou takové, které mají pevně naprogramovaný neměnící se obsah. Dynamické stránky jsou takové, které se generují dynamicky. To znamená, že server uživateli poskytuje pouze to, co zrovna potřebuje. Elektronický dotazník využívá dynamický typ webových stránek. Stránky se přizpůsobují typu uživatele. Jak bude zmíněno v kapitole 7, aplikace rozlišuje tři uživatelské účty (viz kap. 7.2): pacient, doktor a administrátor. Prohlížeč generuje žádost a posílá ji na server. Ten na základě typu uživatelského účtu generuje šablonu uživatelského rozhraní, do které vloží data z databáze, a odesílá ji klientovi.

6.2 Node.js

Pro vývoj logické části bakalářské práce bylo použito prostředí Node.js. Je to open-source prostředí, které slouží k programování serverových částí v jazyce JavaScript. Je postavený na Chrome V8 enginu napsaném v C++. Tento engine zde vystupuje v roli překladače z JavaScript kódu do kódu strojového.

6.2.1 Architektura Node.js

V Node.js je kladen důraz na vysokou škálovatelnost, umožňuje obsluhu mnoha připojených klientů současně. Je to zajištěno pomocí technologie asynchronních a neblokujících se vstupů a výstupů (I/O). Tato technologie umožňuje serveru neustále zpracovávat požadavky klientů, aniž by klient muset čekat na zpracování dříve odeslaných požadavků [30].

Node.js využívá tzv. jednovláknovou architekturu se smyčkou událostí (Single Threaded with Event Loop). Kroky této architektury jsou popsány níže:

1. Do smyčky vstupují požadavky uživatelů tzv. události (event).
2. Tyto události jsou následně přiděleny jednotlivým nezávislým vláknům a umístěny do fronty, tzv. fronty událostí.
3. V další části web serveru, v tzv. smyčce událostí (Event loop), přijímá a zpracovává žádosti. Smyčka událostí používá pouze jedno vlákno.
4. Event Loop poté z fronty událostí vyzvedne jeden požadavek klienta a začne tento požadavek zpracovávat.
5. Pokud proběhne zpracování požadavku bez chyb, připraví odpověď a pošle ji uživateli.

Pro programování serverové části elektronického dotazníku se také využívá pomocný framework Express. Jeho knihovna zjednodušuje programování aplikací, které zpracovávají požadavky HTTP na serveru.

6.2.2 Instalace Node.js

Prvním krokem pro práci s prostředím Node.js je jeho stažení z oficiální stránky `https://nodejs.org/en/download/`. Dalším krokem je instalace balíčku Node.js, inicializace projektu přes příkazový řádek pomocí příkazu „npm init“ a následná instalace modulu Express.js příkazem „npm install <nazev modulu>“.

Dalším důležitým prvkem serverové části je databáze (viz kap. 6.3). Balíček MongoDB je nejprve třeba stáhnout společně s drivery z oficiálních stránek `https://www.mongodb.com/download-center/community` a `https://mongodb.github.io/node-mongodb-native/?jmp=docs`. Následně probíhá instalace a připojení databáze do projektu obdobným způsobem jako tomu bylo u modulu Express.js.

6.3 Databáze

Databáze je systém souborů, které jsou systémově strukturovány tak, aby byly tyto soubory nalezitelné a zpracovatelné. Jinými slovy, databáze je úložiště dat. Zpracování těchto dat zajišťuje tzv. systém řízení báze dat (SŘBD).

Systém řízení báze dat je soubor jazykových a softwarových nástrojů, které umožňují přístup k datům, umožňují jejich vytváření, změnu, mazání a zabezpečení. SŘBD je založen na databázových modelech, které mají různé struktury zpracování dat. Existuje mnoho řešení implementující různé databázové modely. Nejpoužívanější je model relační (RSŘBD).

6.3.1 SQL

Relační model nabízí matematické řešení, jak strukturovat, ukládat a používat data. Vztahy umožňují seskupovat data ve formě tabulek, které jsou mezi sebou propojeny pomocí klíčů: cizího a primárního. Do tabulky jsou ukládány záznamy ve formě řádků. Každý záznam tak tvoří stejný počet položek stejného datového typu. Tento model používá strukturovaný dotazovací jazyk (SQL) pro definování a zpracování dat. SQL dotazy jsou jednoduché lineární posloupnosti operátorů. Každý operátor začíná klíčovým slovem, jako je SELECT, CREATE, INSERT INTO atd. (viz výpis 6.1) [31]. Nejpoužívanější relační SŘBT jsou: MySQL, SQLite a PostgreSQL.

Výpis 6.1: Příklad SQL příkazů pro vytvoření tabulky reprezentující pacienty, jejich přidání a následný výpis

```
CREATE TABLE Patients (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Password varchar(255)
); //založení nové tabulky

INSERT INTO Patients (PersonID, LastName, FirstName, Password)
VALUES ('12062019', 'Příjmení', 'Jméno', 'Heslo'); //přidání nového pacienta

SELECT * FROM Patients; //výpis všech pacientů
```

6.3.2 NoSQL

V dnešní době získává na popularitě nerelační databáze (NoSQL). Tento model poskytuje dynamickou datovou strukturu [32]. Výhody použití dynamické struktury jsou:

- tvorba dokumentů bez nastavování jejich struktury předem,
- každý dokument může mít svou vlastní strukturu,
- každá databáze může mít svou vlastní syntaxi,
- možnost přidání pole přímo při práci s daty.

NoSQL se nejvíce hodí k ukládání velkých objemů dat, u kterých není třeba zachovávat vzájemné vztahy. Mezi NoSQL databázemi jsou nejznámější: CouchDB, MongoDB, Cassandra a HBase. Data v NoSQL mohou být uložena několika způsoby:

- Sloupcově orientované databáze – data jsou uložena ve sloupcové podobě a ne v řádkové, jak je to v relačních databázích.
- Dokumentové databáze – k ukládání dat se používá dokument, do kterého se ukládají páry klíč-hodnota. Dokumenty mohou obsahovat mnoho různých párů klíč-hodnota nebo páry klíč-pole nebo dokonce vložené dokumenty (viz. kap 6.3.3).
- Grafové databáze – používají se hlavně k ukládání informací v sociální sféře, kdy je důležité zaznamenávat vztahy mezi jednotlivými uzly.
- Databáze párové klíč-hodnota – je nejjednodušší způsob uchovávání dat. Každá jednotlivá položka v databázi je uložena jako klíč s jeho hodnotou.

Základní rozdíly NoSQL a SQL:

1. Struktura a datový model – SQL má pevně definovanou strukturu pro ukládání dat. NoSQL má dynamickou strukturu.
2. RSŘBT se řídí požadavky ACID[1] pro transakční[2] systém: atomicity (atomicita neboli nedělitelnost), consistency (konzistence), isolation (izolovanost), durability (trvalost). Naproti tomu NoSQL využívá BASE: basic availability (základní dostupnost), soft state (pružný stav), eventual consistency (eventuální konzistence).
3. Škálovatelnost neboli rozšiřitelnost – rozšíření diskových kapacit na webových serverech. SQL využívá vertikálního škálování – navýšení výkonu fyzického serveru se provádí například přidáním paměti a výměnou procesoru za výkonnější. NoSQL využívá horizontálního škálování, místo vylepšování jednoho serveru se data rozprostírají na více serverech, které spolu vzájemně spolupracují.

Vzhledem k výše uvedeným výhodám a zvyšující se popularitě byla v bakalářské práci použita MongoDB NoSQL databáze.

6.3.3 MongoDB

MongoDB je open-source dokumentová databáze, která k ukládání dat využívá dokumenty. Na rozdíl od řádků u SQL, dokumenty umožňují uchovávat informace, které mají složitou strukturu. Dokument může být reprezentován jako úložiště klíčů a hodnot. Klíč představuje jednoduchou značku, ke které je přiřazena určitá informace. Pokud klíč nemá přiřazenou hodnotu, pak je tento klíč v dokumentu vynechán a není používán. Jednotlivé dokumenty mohou být spojeny do jedné kolekce. Kolekce může obsahovat různé dokumenty, které mají různou strukturu a vlastnosti.

[1] ACID je zkratka oblasti databází. Jde o první písmena čtyř anglických slov vystihujících čtyři žádoucí vlastnosti databázových transakcí.

[2] Transakce je skupina úloh, které je třeba vykonat jako jeden funkční celek.

Příklad dokumentu MongoDB je zobrazen ve výpisu 6.2. Ve výpisu 6.3 je znázorněn příklad dokumentu s uloženým pacientem.

Výpis 6.2: Příklad šablony pro vytvoření uživatelů v MongoDB

```
{
  name: String, //jméno
  surname: String, //příjmení
  id: int, //rodné číslo
  password: String, //heslo
  typeUser: String, //typ uživatele
  parrentDoctor: String, //id doktora, který přidal pacienta
  selectTest: [String] //pole zvolených testy pro příslušného pacienta
}
```

Výpis 6.3: MongoDB dokument s uloženým pacientem

```
 {
    "_id": {
        "$oid": "5cd9817f6fe5747c26891a7d" //id dokumentu
    },
    "selectTest": [
        "5cd97e226fe5747c26891a24", // id zvolených testů
        "5cd97f3f6fe5747c26891a6f"
    ],
    "name": "Patient", //jméno
    "surname": "#1", //příjmení
    "id": "3", //rodné číslo
    "password": "$2b$10$h1eO73H61HNW/N0LRyVpkuhQpOC2pFZV1YeDrwbfrsk6/rCd.gnNS",
    //zahašované heslo
    "typeUser": "pacient", //typ uživatele
    "parrentDoctor": "5cd97e9a6fe5747c26891a3d",
    //id doktora, který přidal příslušného pacienta
}
```

Jednotlivé dokumenty jsou uložené ve formátu BSON. Je to binárně reprezentovaný JSON formát. JSON (JavaScript Object Notation) je způsob zápisu dat ve formě páru klíč-hodnota. JSON umí uchovávat objekty, textové řetězce, čísla, pole hodnot, datumy atd. BSON umožňuje pracovat s daty rychleji: rychlejší vyhledávání a zpracování. Každý dokument obsahuje unikátní identifikátor. MongoDB tento identifikátor generuje sama, lze ho však zadat i ručně. Automaticky vygenerovaný identifikátor má 12 bajtů. Ukázkový identifikátor je zobrazen ve výpisu 6.4 a skládá se z: 4-bajtů představujících sekundy od Unix epochy, 5-bajtů – náhodného čísla a 3-bajtového počítadla startujícího na náhodné hodnotě [33].

Výpis 6.4: Příklad identifikátora dokumentu v MongoDB

```
5cd9817f6fe5747c26891a7d
```

7 Implementace klientské části systému

V této kapitole bude prezentována vyvinutá mobilní webová aplikace, použité technologie a detaily vývoje klientské části systému. Mobilní aplikace elektronického dotazníku byla vyvinuta jako webová stránka. Psaní kódu probíhalo v textovém editoru Atom, který urychlil psaní pomocí nainstalovaných pomocných pluginů (emma).

Vývoj webové aplikace probíhal s využitím programovacích jazyků, jako jsou HTML, CSS a JavaScript, jejichž podrobný popis je popsán v článku 3.2.1. K usnadnění vývoje byly použity tyto pomocné technologie:

- framework UIKit pro tvorbu designu webu,
- knihovnu React.

7.1 Použité technologie klientské části

7.1.1 UIKit

Pro tvorbu designu byl použit framework UIKit. Framework je sada nástrojů, knihoven a hotových modulů, které využívá vývojář při tvorbě webových stránek. Výhody použití frameworků jsou:

- Zjednodušuje a urychluje proces vývoje. Místo toho, aby si vývojář sám psal HTML kód a následně ho graficky ladil, může si otevřít databázi komponentů příslušného UI frameworku a prostým kopírováním si hledaný komponent vložit do svého projektu.
- Obsahují téměř vše, co je potřeba, od jednoduchých css-stylingových textů přes javascriptové komponenty modálních oken, záložek, posuvníků, tabulek, tlačítek atd.
- Umožňují vytvářet adaptivní webové stránky. Stránky se přizpůsobují různým rozlišením obrazovky: notebooky, tablety, telefony.

UIKit je framework ve stylu Material Design, který má ve své databázi kolem 30 flexibilních a adaptivních komponentů. Pro práci s frameworkem je potřeba si ho stáhnout z oficiálních stránek do svého počítače a následně ho připojit v hlavičce HTML souboru [28]. Seznam komponentů a stručný popis frameworku lze najít na `https://getuikit.com/`.

7.1.2 JavaScriptova knihovna React

React je open – source nástroj pro tvorbu uživatelského rozhraní. Je to nová knihovna, která byla vyvinuta Facebookem a anoncována v roce 2015. Jejím hlavním úkolem je zobrazování obsahu HTML stránky na display zařízení. React usnadňuje tvorbu

uživatelského rozhraní rozdělením obsahu stránky do malých částí. Těmto částem se říká komponenty (tlačítka, navigační panely, registrační pole, textové řetězce atd.). Každý komponent je JavaScriptová funkce, která představuje určitý fragment webové stránky. Následně se tyto funkce vyvolávají v určitém pořadí, sdružují se a zobrazují na obrazovce. Další výhodou je, že komponenty mohou být znovu použity v originální nebo i pozměněné podobě.

Pro psaní kódu v textovém editoru se používá JSX. JSX (JavaScript XML) je rozšíření syntaxe JavaSciptu, které umožňuje použít syntaxi podobnou jazyku HTML k popisu struktury uživatelského rozhraní. Tuto vlastnost zajišťuje kompilátor Babel. Je to kompilátor, který překládá kód napsaný značkovacím jazykem do JavaScriptu.

Dalším důvodem pro použití React knihovny je rychlost aplikace. Výkon takové aplikace vysoce převyšuje ty aplikace, které jsou vyvíjeny pouze s využitím JavaScriptu. Hlavním důvodem této rychlosti je technologie Reactu s názvem Virtual DOM [29].

Virtual DOM

DOM (Document Object Model) je způsob skládání strukturovaných dokumentů z objektů. Jedná se o multiplatformní a jazykově nezávislou prezentaci HTML nebo XML dokumentů. Vývojář může pracovat s uzly dokumentů, měnit jejich data, mazat je nebo vkládat nové uzly. Nevýhodou DOMu je pomalé zpracování dynamického uživatelského rozhraní a obsáhlých webových stránek, jelikož obnovení všech uzlů může zabrat dost času.

K řešení tohoto problemu se využívá technologie virtuálního DOMu. Virtual DOM není standard a v konečném důsledku vývojář pracuje s DOMem. Dělá to ale zřídka a efektivněji. Případné změny DOMu se nejprve provedou v DOMu virtuálním. Systém pak porovnává tyto dvě verze a ve skutečném DOMu pozmění jen ty části, které se liší. Tento postup je rychlejší, protože systém nemusí přímo pracovat s velkým množstvím dat reálného DOMu. Existují dvě metody porovnávání dat:

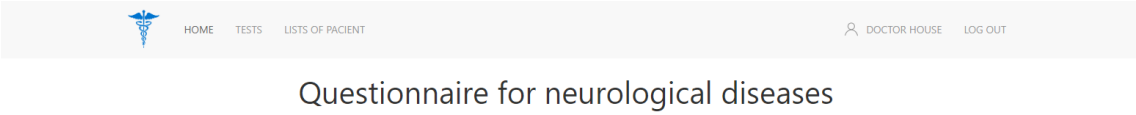
- Hrubá kontrola (Dirty Cheking) – v pravidelných intervalech odesílá dotazy na změny dat.
- Pozorování (Observable) – jde o sledování změny stavu. Pokud se nic nemění, data se neaktualizují.

7.2 Uživatelské rozhraní

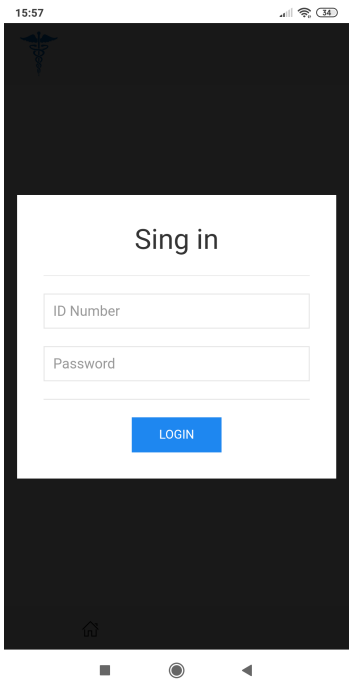
Výhody progresivní webové aplikace spočívají v tom, že může být použita jak v počítači jako webová stránka, tak třeba na telefonu nebo tabletu jako mobilní aplikace.

Uživatel si může zvolit, které zařízení pro práci s dotazníkem použije.

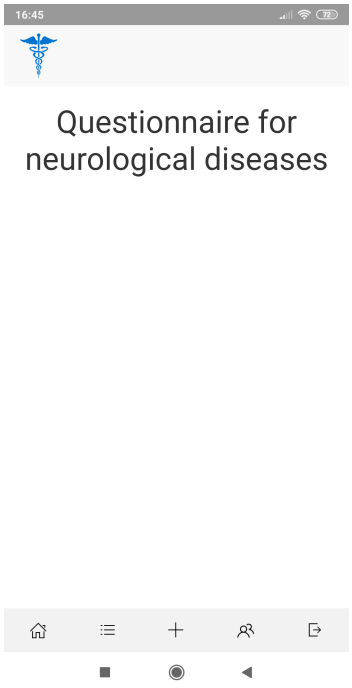
Aplikace využívá adaptivní design. Díky tomu se automaticky přizpůsobí velikost obrazovky zařízení. Může pracovat v režimu na šířku i na výšku. Obrázek 7.1 a obrázek 7.3 ukazuji rozdíl mezi počítačovým rozhraním a mobilním rozhraním. V mobilním rozhraní se navigační panel nachází zespodu obrazovky. Navigační panel obsahuje tlačítka odpovídající záložkam webové stránky.



Obr. 7.1: Počítačové uživatelské rozhraní



Obr. 7.2: Přihlašovací okno



Obr. 7.3: Mobilní rozhraní

Navigační panel obsahuje takové záložky jako: úvodní stránka „Home“, seznam

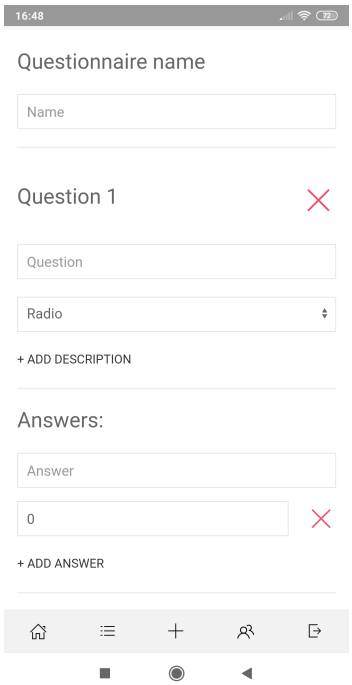
dostupných dotazníků „Tests“ , výsledky testů „My Results“, záložku, která je k dispozici pouze pro lékaře, „List of Patients“ a také jméno přihlášeného uživatele a tlačítko odhlášení. Jelikož je aplikace zabezpečená, bude po spuštění aplikace vyžadována autorizace. Uživatel se bude muset přihlásit ke svému účtu, jak je znázorněno na obr. 7.2.

Elektronický dotazník umožňuje přihlášení třemi typy uživatelských účtů: administrátor, doktor a pacient. Každý typ účtu se odlišuje vlastním uživatelským rozhraním a různými oprávněními. O jaký typ účtu se jedná a jaké uživatelské rozhraní vygenerovat a zobrazit rozhoduje po přihlášení uživatele do aplikace server s databází.

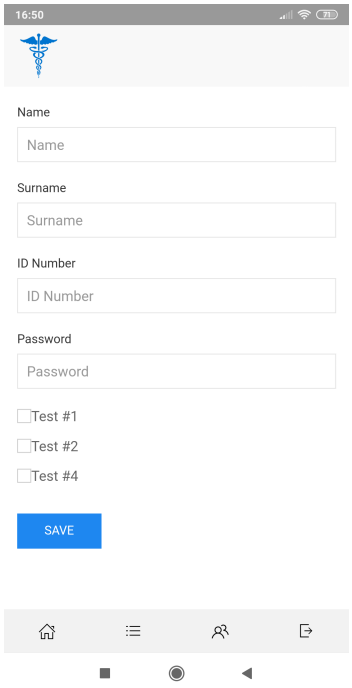
7.2.1 Rozhraní administrátorské

Uživatelský účet administrátora existuje pouze jeden. Administrátora nelze vytvořit přes uživatelské rozhraní. Je vytvořen přímo ve zdrojovém kódu a uložen do databáze. Administrátor je především odpovědný za tvorbu, změnu a mazání testů pomocí příslušné šablony, která je zobrazena na obr. 7.4. Otázky nabízí dva typy odpovědí:

- Radiobutton – z nabízených možností lze zvolit pouze jednu variantu.
- Slovní odpověď – je slovně zodpovězená odpověď o libovolném počtu znaků.

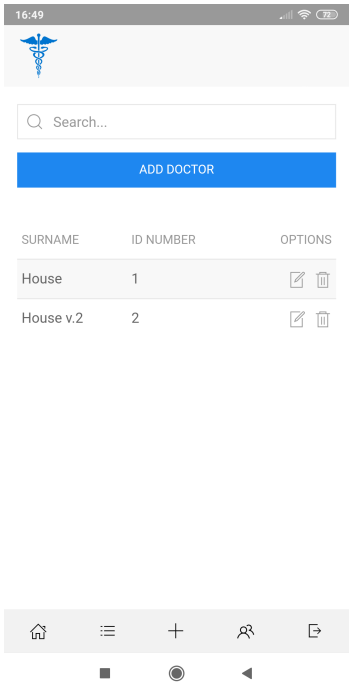


Obr. 7.4: Založení nového testu

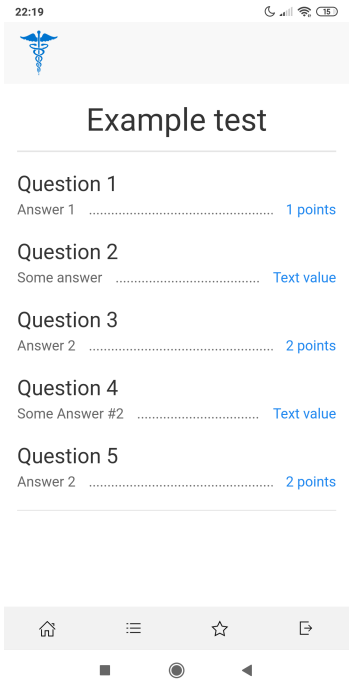


Obr. 7.5: Přidávání pacienta

Další funkcí, kterou disponuje účet administrátora, je možnost tvorby, editace nebo mazání doktorských účtů. Pro přehlednost jsou účty zobrazeny ve formě tabulky s možností vyhledávání podle jména. Seznam doktorů je zobrazen na obr. 7.6.



Obr. 7.6: Seznam doktorů



Obr. 7.7: Výsledek testu

7.2.2 Rozhraní pro doktory

Grafické rozhraní pro doktory se od účtu administrátora liší tím, že doktor nemůže dotazníky ani tvořit, ani měnit, ani mazat. Doktor si tyto testy může pouze zobrazit. Hlavní funkcí doktorských účtů a doktorů je tedy přidávání nových pacientů do systému a sledování jejich výsledků. Každý doktor spravuje a má k dispozici pouze výsledky pacientů, které sám přidal. Při zakládání nového účtu pro pacienta uvádí tyto údaje:

- jméno,
- příjmení,
- rodné číslo, které slouží jako přihlašovací jméno uživatele a je unikátní,
- heslo, které je z bezpečnostních důvodů zobrazeno jako tečky,
- a přidává k účtu požadované dotazníky.

Šablona pro zakládání nových účtů pacientů je zobrazena na obr. 7.5 a dokument, do kterého je uložený záznam o pacientovi v databázi, je zobrazený ve výpisu

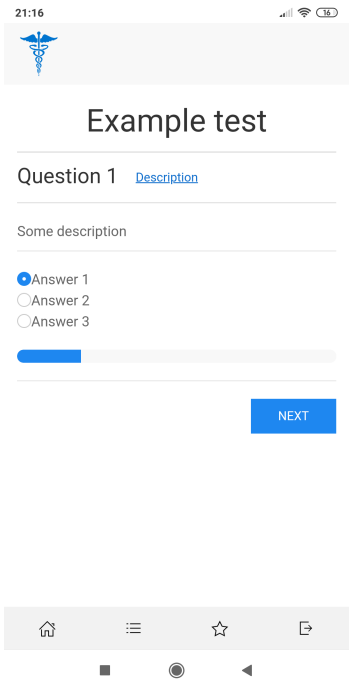
6.3. Výsledky zodpovězených dotazníků jednotlivých pacientů si doktor může zobrazit, editovat, mazat a nebo také exportovat ve formátu .xls. Příklad exportovaného souboru je na obr. 7.8.

	A	B	C	D	E	F	G	H	I	J	K
1	Name pacient	Surname pacient	ID number	Name test	Date	Question 1	Question 2	Question 3	Question 4	Question 5	
2	Patient	#1	01012001 /0001	Example test	23.04.2019 22:12	Answer 1	Some answer	Answer 2	Some Answer #2	Answer 2	
3											
4											

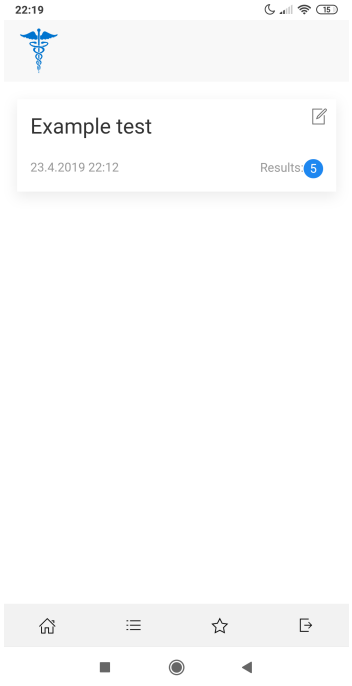
Obr. 7.8: Ukázka exportovaného souboru Excel

7.2.3 Rozhraní pacienta

Pacient má ve svém rozhraní pouze dvě karty „Tests“ a „Results“. V první záložce bude mít k dispozici veškeré testy, které mu doktor zpřístupnil. Ukázka testu je na obr. 7.9. Testové otázky jsou rozloženy tak, že je vždy jen jedna otázka na stránce. Mezi otázkami lze přepínat tlačítky „Next“ a „Previous“. Dále disponuje aplikace tzv. progress barem, což je pomocný ukazatel pacientova progresu v testu. Zobrazuje, kolik otázek pacient už zodpověděl a kolik mu jich ještě zodpovědět zbývá.



Obr. 7.9: Ukázka testu



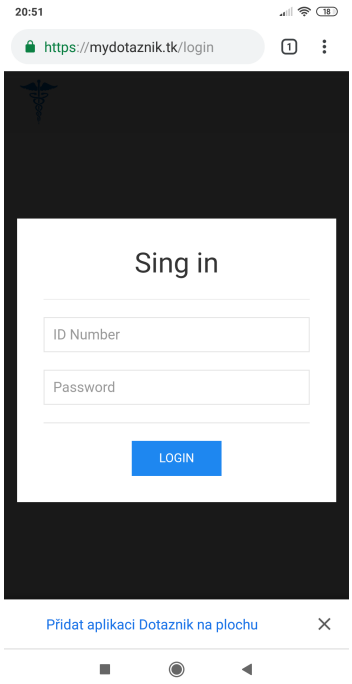
Obr. 7.10: Výsledky všech testů

Pokud pacient test nedokončí, test se i přesto uloží do karty „Results“. Odtud je i s odstupem času možné test opět spustit a doplnit zbývající odpovědi. Ukázka

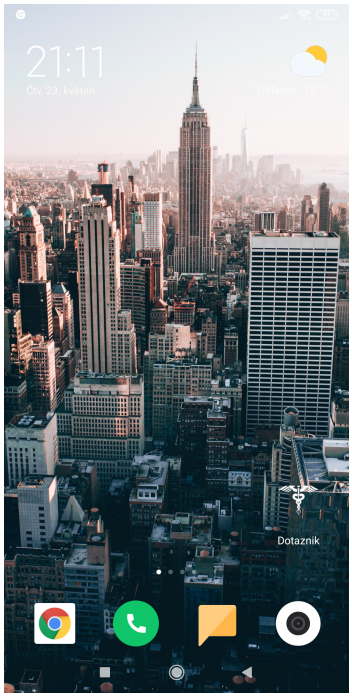
záložky s výsledky je zobrazena na obr. 7.10. Všechny dokončené testy se dále ukládají s informacemi o datu zahájení testu a jeho výsledku. Uložené výsledky lze rozkliknout a prozkoumat (obr. 7.7)

7.3 Režim Offline

Nejprve si uživatel musí do telefonu stáhnout aplikaci. Progresivní webová aplikace nabídne své stažení komukoli, kdo navštíví její internetovou stránku v mobilu nebo tabletu s androidem (viz obr. 7.11). Pokud používá uživatel k prohlížení systém iOS, bude si muset sám aplikaci sdílet na pracovní plochu. Následně se v telefonu zobrazí ikona dotazníku (viz obr. 7.12), která umožňuje jeho rychlé spuštění.



Obr. 7.11: Instalace aplikace na plochu mobilu



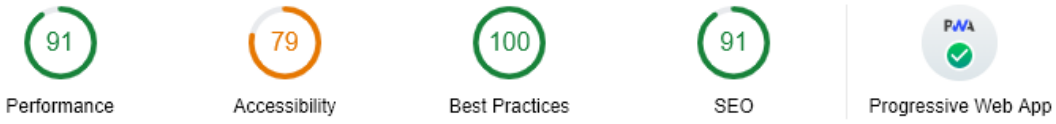
Obr. 7.12: Přidána ikona dotazníku

Úplně offline s aplikací pracovat nelze. Uživatel musí být připojený k síti se serverem minimálně na přihlášení do systému a stažení nových dotazníků. Uživatel se přihlašuje pomocí rodného čísla a hesla. Jakmile je přihlášen, může pacient procházet a odpovídat na stažené dotazníky a ukládat výsledky i offline. Veškerá data potřebná k zobrazení uživatelského rozhraní se ukládají do Cache paměti, která je popsána v kapitole 2.3. Dynamická data jako jsou: přihlášený uživatel, testy a výsledky testů se ukládají do paměti prohlížeče, tzv. LocalStorage. LocalStorage si lze představit jako objekt, který nemaže hodnoty v něm uložené ani tehdy, je-li webová stránka

aktualizována a dokonce ani tehdy, je-li prohlížeč zavřen. Kapacita LocalStorage je kolem 10 MB. Po obnovení připojení k síti se data se serverem automaticky synchronizují.

7.4 Testování

Progresivní webová aplikace byla otestována pomocí nástroje Lighthouse (viz kap. 2.2.5). Výsledek testování aplikace je zobrazen na obrázku 7.13. Všechny položky kromě Accessebility (přístupnost) mají výborné výsledky. Důsledkem klesání této položky je bílé pozadí aplikace, které žádným způsobem neovlivňuje funkčnost celého systému.



Obr. 7.13: Lighthouse

Mobilní aplikace byla také nainstalována a otestována na mobilním zařízení. V operačním systému Android a iOS testování proběhlo úspěšně.

8 Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat aplikaci sloužící k rychlému vyšetření pacientů pomocí dotazníků. Aplikace má být založena na technikách mHealth a má fungovat na všech nejpoužívanějších platformách a zařízeních. Z důvodu ochrany osobních údajů pacientů má být zabezpečena.

V teoretické části bakalářské práce jsou důkladně rozebrány základní typy mobilních aplikací a je posouzena jejich vhodnost pro elektronický dotazník v závislosti na požadavcích a dostupných technologiích. Dále je nastíněna problematika zabezpečení a je zdůvodněn konečný výběr technologie PWA pro tvorbu aplikace. Tato technologie je dále detailněji popsána i spolu s nástroji, které využívá. Dále je podrobně popsána komunikace probíhající mezi klientem a serverem a použité zabezpečovací prvky zajišťující ochranu osobních údajů uživatelů. V závěru teoretické části je detailně rozebrána serverová a klientská část aplikace, jsou popsány technologie a frameworky ve vývoji použité a je zde detailní popis grafického rozhraní.

V rámci praktické části proběhla realizace vývoje progresivní webové aplikace tak, aby splňovala všechny požadované vlastnosti zabezpečeného elektronického dotazníku. Dokončená webová aplikace je nyní zveřejněna na neplaceném webovém hostingu a byla ji přiřazena doména mydotaznik.tk.

Konečný produkt byl otestován jak na počítači jako webová stránka, tak i na mobilních zařízeních běžících na operačních systémech Android a iOS. Ve všech testovaných prohlížečích (Safari, Google Chrome a Mi prohlížeč) aplikace pracovala spolehlivě a plnila všechny požadavky na ni kladené. Jediným nedostatkem aplikace je omezená kapacita paměti využitelné v offline režimu.

Literatura

[1] What is mHealth Technology. Ausmed [online]. b.r. [cit. 2018-10-02]. Dostupné z: https://www.ausmed.com/articles/what-is-mhealth/

[2] Neurological Diagnostic Tests and Procedures. National institute of neurological disorders [online]. b.r. [cit. 2018-10-02]. Dostupné z: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Neurological-Diagnostic-Tests-and-Procedures-Fact

[3] Encryption. Search security [online]. b.r. [cit. 2018-10-02]. Dostupné z: https://searchsecurity.techtarget.com/definition/encryption

[4] DUCKETT, J. *HTML and CSS: Design and Build Websites*, 2011, vol. 1, John Wiley and Sons, Inc., Indianapolis, Indiana 511 s ISBN: 978-1-118-00818-8.

[5] DUCKETT, J. *JavaScript and JQuery: Interactive Front-End Web Development*, 2014, vol. 1, John Wiley and Sons, Inc., Indianapolis, Indiana 645 s ISBN: 978-1-118-00818-8.

[6] What are the popular types and categories of apps. Mobile technologies [online]. b.r. [cit. 2018-10-02]. Dostupné z: https://thinkmobiles.com/blog/popular-types-of-apps/

[7] Progressive Web Apps. Google Developers [online]. b.r. [cit. 2018-10-03]. Dostupné z: https://developers.google.com/web/progressive-web-apps/

[8] Web Fundamentals. Google Developers [online]. b.r. [cit. 2018-10-03]. Dostupné z: https://developers.google.com/web/fundamentals/?hl=en

[9] Web Worker. Mozilla Developers [online]. b.r. [cit. 2018-10-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

[10] Tools for Web Developers. Google Developers [online]. b.r. [cit. 2018-10-03]. Dostupné z: https://developers.google.com/web/tools/lighthouse/

[11] Application Shell. Google Developers [online]. b.r. [cit. 2018-10-09]. Dostupné z: https://developers.google.com/web/updates/2015/11/app-shell

[12] Manifest. Google Developers [online]. b.r. [cit. 2018-10-09]. Dostupné z: https://developers.google.com/web/fundamentals/web-app-manifest/

[13] Memory. Google Developers [online]. b.r. [cit. 2018-10-09]. Dostupné z: https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa

[14] Push Notifications. Google Developers [online]. b.r. [cit. 2018-10-09]. Dostupné z: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications

[15] Service Worker. Google Developers [online]. b.r. [cit. 2018-10-10]. Dostupné z: https://developers.google.com/web/fundamentals/primers/service-workers/

[16] HTTPS. Google Developers [online]. b.r. [cit. 2018-10-10]. Dostupné z: https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https

[17] Secure. TipTopSecurity [online]. b.r. [cit. 2019-16-04]. Dostupné z: https://tiptopsecurity.com/how-does-https-work-rsa-encryption-explained/

[18] MD5. MIT Laboratory for Computer Science and RSA Data Security [online]. b.r. [cit. 2019-18-04]. Dostupné z: https://tools.ietf.org/html/rfc1321

[19] Rainbow table. GeeksForGeeks Computer science portal [online]. b.r. [cit. 2019-18-04]. Dostupné z: https://www.geeksforgeeks.org/understanding-rainbow-table-attack/

[20] Bcrypt. NPM open-source products [online]. b.r. [cit. 2019-19-04]. Dostupné z: https://www.npmjs.com/package/bcrypt

[21] JEŘÁBEK, J. *Komunikační technologie: Elektronická skripta*, 1. vydání Brno: VUT Brno, 2013, ISBN 978-80-214-4713-4.

[22] GOURLEY, D., TOTTY, B. *HTTP:The Definitive Guide*, 1. vydání O'Reilly Media, 2002, 658 s ISBN-10: 9781565925090.

[23] Web hosting. Namecheap Inc. [online]. b.r. [cit. 2018-10-19]. Dostupné z: https://www.namecheap.com/hosting/what-is-web-hosting-definition/

[24] Text Editor. Designrevision [online]. b.r. [cit. 2018-10-19]. Dostupné z: https://designrevision.com/best-code-editor/

[25] HTML tutorial. HTML.com: Study HTML and Learn To Code [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://html.com/

[26] HTML tutorial. W3schools tutorials [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://www.w3schools.com/hTml/html_head.asp

[27] HTML5 tutorial. HTML5 learning courses [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://www.tutorialspoint.com/html5/

[28] UIKit. UIKit dokumentace [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://getuikit.com/

[29] STEFANOV, S. *ReactJS Up and Running*, 1. vydání O'Reilly Media, 2016, 222 s ISBN: 978-1-491-93182-0.

[30] BROWN, E. *Web Development with Node and Express*, 1. vydání O'Reilly Media, 2014, 222 s ISBN: 978-1-491-94930-6.

[31] SQLCourses. Online SQL Training [online]. b.r. [cit. 2019-04-25]. Dostupné z: http://www.sqlcourse.com/

[32] Pillar Global. THE DIFFERENT TYPES OF NOSQL [online]. b.r. [cit. 2019-04-25]. Dostupné z: https://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases

[33] MongoDB. MongoDB dokumentace [online]. b.r. [cit. 2019-04-25]. Dostupné z: https://www.mongodb.com/

[34] CSS tutorial. CSS learning portal [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://www.csstutorial.net/

[35] Backend. Simple programmer [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://simpleprogrammer.com/what-is-back-end-development/

[36] JavaScript tutorial. JavaScript learning portal [online]. b.r. [cit. 2018-11-16]. Dostupné z: https://javascript.info/

[37] FLANAGAN, D. *JavaScript: The Definitive Guide*, 2011, vol. 6, United States of America by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol 1096 s ISBN: 978-0-596-80552-4.

Seznam symbolů, veličin a zkratek

API	Application Programming Interface – sada metod, protokolů a tříd pro programování
CSS	Cascading Style Sheets – kaskádové styly
DNS	Domain Name System – hierarchický systém doménových jmen
DOM	Document Object Model – způsob skládání strukturovaných dokumentů z objektů
FTP	File Transfer Protocol – internetový protokol pro přenos souborů
GPS	Global Positioning System – globální polohový systém
HTML	Hypertext Markup Language – značkovací jazyk
HTTP	Hypertext Transfer Protocol – internetový protokol pro přenos hypertextových dokumentů
HTTPS	HyperText Transfer Protocol Secure – zabezpečený internetový protokol pro přenos hypertextových dokumentů
IMAP	Internet Message Access Protocol – internetový protokol pro vzdálený přístup k e-mailové schránce
iOS	iPhone OS – mobilní operační systém
IP	Internet Protocol – internetový protokol
JSON	JavaScript Object Notation – způsob zápisu dat
JSX	JavaScript XML – je rozšíření syntaxe JavaSciptu
PHP	Hypertext Preprocessor – skriptovací programovací jazyk
POP3	Post Office Protocol – internetový protokol pro stahování e-mailových zpráv
PWA	ProgressiveWeb Application – progresivní webové aplikace
SMTP	Simple Mail Transfer Protocol – internetový protokol pro přenos zpráv elektronické pošty
SQL	Structured Query Language – strukturovaný dotazovací jazyk
SSL	Secure Sockets Layer – kryptografický protokol
SW	Service worker – mechanismus, který spouští skripty na pozadí
TLS	Transport Layer Security – kryptografický protokol
URL	Uniform Resource Locator – řetězec znaků s definovanou strukturou
XHTML	eXtensible Hypertext Markup Language – rozšiřitelný hypertextový značkovací jazyk
XML	eXtensible Markup Language – rozšiřitelný značkovací jazyk

Seznam příloh

A Obsah přiloženého CD **54**

A Obsah přiloženého CD

Přiložené CD obsahuje soubory vytvořené v rámci bakalářské práce:

- Bakalářská práce (.pdf)
- Zdrojové kódy (.zip)