Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Systém pro realizaci elektronických anket

Bakalářská práce

Autor: Tomáš Rýzner
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph.D.

Odborný konzultant: Jan Charamza
ŠKODA AUTO

Hradec Králové říjen 2022

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 14.8.2022 Tomáš Rýzner

Poděkování:
Děkuji vedoucímu bakalářské doc. Mgr. Tomáši Kozlovi Ph.D. za vedení práce, cenné připomínky a trpělivost při její realizaci.

Anotace

Tato práce shrnuje proces návrhu a vývoje webové aplikace určené pro vytváření a vyplňování dotazníků. Zadání pro tvorbu aplikace poskytla firma ŠKODA AUTO. Cílem bylo vytvořit aplikaci, která uživateli umožní nenáročné vytvoření elektronického dotazníku a získání zpětné vazby. Práce obsahuje výběr a popis jednotlivých technologií použitých pro tvorbu aplikace a popis implementace konkrétních technologií. Úvodní část práce obsahuje zadání od firmy ŠKODA AUTO. Hlavní část práce se zabývá výběrem technologií, jejich popisem a implementací. Závěrečná část práce se věnuje využití aplikace v praxi a získaným datům.

Annotation

Title: System for realization of electronic surveys

This work summarizes the process of design and development of web application intended for creation and filling of surveys. Assignment for creation of the app was provided by company ŠKODA AUTO. The goal was to create an application, that will allow the user a simple creation of an electronic survey and getting feedback. The work contains the selection and description of individual technologies used for creation of the application and the description of implementation specific technologies. The introduction contains the assignment from ŠKODA AUTO. The main part of the work deals with the selection of technologies, their description, and their implementation. The final part of the work is devoted to the use of the application in practice and gained data.

Obsah

Seznam obrázků

1 Úvod

Pro oddělení předvývoje spolčnosti Škoda AUTO je zpětná vazba velmi důležitou součástí. Dokáže projektovému manažerovi poskytnout cenná data ohledně daného projektu a upozornit na chyby či na aspekty, které v projektu chybí. V oddělení předvývoje, kde jsem strávil 3 roky na pozici stážisty, se zpětná vazba od uživatelů získávala pomocí tištěných dotazníků. Sběr dat z takto připravených dotazníků byl poté velmi časově náročný. Obecně se jednalo o několik hodin přepisování dat do excelových tabulek. Mým úkolem tedy bylo vytvořit aplikaci, která by urychlila jak vytváření dotazníků, tak jejich finální vyhodnocení.

Jednou z možností využití aplikace je vytváření individuálních dotazníků podle potřeby jednotlivých zaměstnanců oddělení. Další možností je využití aplikace pro veletrh inovací s názvem IVET, který každoročně oddělení předvývoje pořádá. Jedná se o soukromou akci jak pro představenstvo firmy ŠKODA AUTO, tak pro interní zaměstnance. Cílem akce je představení nových inovací vyvinutých oddělením předvývoje a pokrok vývoje na inovacích, které byly představeny během minulých ročníků této akce.

2 Sběr a analýza požadavků

Pro rychlý a efektivní vývoj aplikace je zapotřebí podrobný soupis požadavků. Zpočátku vývoje aplikace existoval pouze hrubý soupis požadavků, protože nebylo zřejmé, které funkce by měla aplikace poskytovat. Požadavky na konkrétní funkcionalitu přibývaly až s prvními verzemi aplikace, a to od vedoucího stáže nebo samotných uživatelů.

2.1 Zabezpečení dat

Protože výsledná aplikace bude přístupná z internetu, je zapotřebí zabezpečení dat. Požadavky na zabezpečení dat:

- Zabezpečení obsahu pomocí přihlášení
- Zabezpečení přístupu k dotazníkům
- Zabezpečení serveru využívaného pro hosting

2.2 Tvorba dotazníků

Každý uživatel má konkrétní požadavky na funkcionalitu aplikace. Během vývoje byla zvážena implementace každé požadované funkcionality. Cílem bylo poskytnout uživatelům co největší množství funkcí, ale zároveň zachování jednoduchého a intuitivního uživatelského rozhraní.
Požadavky na tvorbu dotazníků:

- Tvorba textových otázek
- Možnost vkládání obrázků k otázkám
- Možnost volby z předem definovaných odpovědí

2.3 Zobrazení výsledků

Po získání zpětné vazby od respondentů je zapotřebí uživateli zobrazit výsledky dotazníku pomocí grafů.
Požadavky na zobrazení výsledků:

- Zobrazení výsledků dotazníku pomocí grafů
- Zobrazení odpovědí jednotlivých respondentů

- Možnost exportu výsledků do excelového souboru

2.4 Uživatelské prostředí

Důležitou součástí každé moderní aplikace je intuitivní uživatelské rozhraní. Jelikož cílem aplikace bylo umožnit respondentům vyplnění dotazníku z jejich telefonu, je také zapotřebí zajistit responzivitu aplikace.

Požadavky na uživatelské prostředí:

- Responzivita aplikace pro mobilní zařízení
- Přívětivé a intuitivní prostředí

2.5 Inovační veletrh

Oddělení předvývoje každoročně uspořádává inovační veletrh, kde představuje nové automobilové inovace. V rámci inovačního veletrhu bylo plánováno využití aplikace pro sběr zpětné vazby od účastníků.

Požadavky na funkcionalitu pro inovační veletrh:

- Možnost vstupu do dotazníku pomocí NFC tagu
- Jazykové mutace dotazníků
- Zvolení jazyka dotazníku podle nastaveného jazyka prohlížeče
- Stabilita aplikace při vysokém vytížení

2.6 Uživatelské role a interakce se systémem

Na základě výše zmíněných požadavků lze dedukovat, jakým způsobem bude uživatel se systémem pracovat. Protože je vyžadováno zabezpečení systému, je zapotřebí definovat uživatelské role. Jednotlivé uživatelské role definují, jaké akce bude uživatel schopný v rámci systému provádět.
Seznam uživatelských rolí:

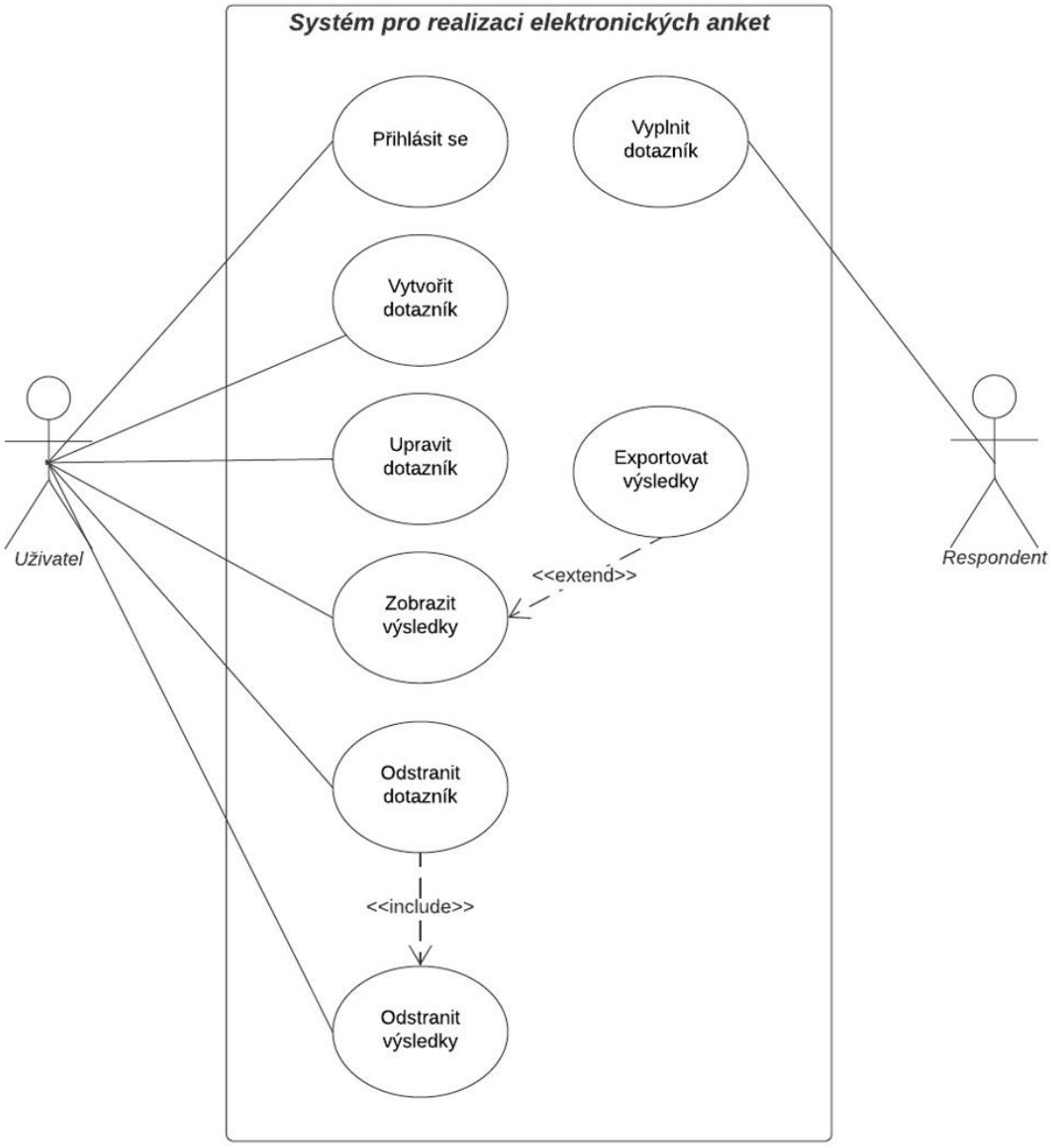
- Uživatel
- Respondent

2.6.1 Uživatel

Uživatel bude moct po registraci vytvářet dotazníky a podle potřeby je upravovat či mazat. Po získání zpětné vazby bude mít možnost zobrazit výsledky jednotlivých dotazníků, případně exportovat výsledky do excelové tabulky.

2.6.2 Respondent

Jediná akce, kterou bude respondent v rámci systému moct provádět, je vyplnění konkrétního dotazníku.



Obr. 1 Use case diagram
Zdroj: vlastní zpracování

3 Problematika tvorby webových aplikací

Protože ze strany společnosti ŠKODA AUTO nebyly uvedené žádné požadavky na použité technologie, byl výběr technologií v režii autora. V dnešní době existuje nespočet technologií pro vytváření webových aplikací. Cílem tedy bylo vybrat rychlé a spolehlivé technologie, pomocí kterých lze aplikaci efektivně vyvinout.

3.1 Historie webových stránek

3.1.1 První stránky

První webové stránky vznikly v 90. letech. V této době byla většina stránek pouze vzájemně propojené HTML dokumenty. Stylování a pozicování HTML elementů bylo řešeno pomocí HTML tagů a veškerý obsah byl statický (1).

3.1.2 CSS

Jedním z problémů prvních webových stránek byl nedostatečné možnosti úpravy obsahu. Tento problém vyřešil příchod kaskádových stylů, které oddělují strukturální jazyk (HTML) od vizuálního (CSS). CSS umožňuje úpravu stylů elementu jako je např. velikost, barva nebo pozice (1).

3.1.3 JavaScript

V roce 1995 vyvinul Brendan Eich programovací jazyk JavaScript (původně "Mocha"), který běží v prohlížeči. Cílem Javascriptu bylo udělat webové stránky interaktivní pomocí automatické úpravy obsahu na základě uživatelského vstupu. Javascript například umožnil upozornění uživatele na nesprávně vyplněný formulář, a to bez potřeby odeslání formuláře serveru (1).

3.1.4 Webové stránky na telefonech

Velký dopad na vývoj webových stránek mělo přestavení telefonu iPhone od společnosti Apple v roce 2007 a následující vlna chytrých telefonů. Zatímco doposud se na web přistupovalo přes počítače, většina nových telefonů měla vlastní webový prohlížeč. Od roku 2007 počet uživatelů navštěvující web z mobilních telefonů

rapidně rostl. Během této doby se musely všechny webové stránky telefonům přizpůsobit (1).

3.1.5 Single-page aplikace

V dnešní době na vývoj webových aplikací používají frontend frameworky, které komunikují s uživatelem pomocí dynamického přepisování obsahu novými daty, namísto původní metody načítání celých webových stránek (1). Jeden z takovýchto frontend frameworků byl využit pro realizaci vyvíjené aplikace.

3.2 Web development stack

Dobrým počátečním bodem je pohled na již vyvinuté aplikace a jaké technologie využívají. Takzvaný „web development stack“ je kolekce softwaru pro tvorbu kompletní platformy, která nepotřebuje další podpůrný software. Může se jednat o jakoukoliv kolekci vzájemně nesouvisejících technologií, které dohromady vytvářejí spolehlivé a plně funkční softwarové řešení. Full development stack se obecně skládá z alespoň tří částí. Jedná se o frontendový framework, backend a databázi (2).

3.3 Frontend vs. backend

Frontend se zaměřuje na uživatele. Jedná se o vytváření elementů a funkcí stránky, které budou viděny uživatelem. Cílem je zajistit správnou funkcionalitu vizuálních prvků stránky. Frontend je také označován jako „klientská strana“ aplikace. Protože vyvíjená webová aplikace bude dynamická, úlohou frontendu je aktualizace dat na stránce.

Na rozdíl od frontendu je backend část webové aplikace, kterou uživatel nevidí. Díky backendu je webová aplikace interaktivní. Hlavní úlohou backendu je obsluha požadavků a komunikace s databází. Backend je také označován jako „serverová strana“ aplikace (3).

3.4 Frontend framework

Frontend framework je předem napsaný kód, který funguje jako základ pro aplikaci. Umožňuje využití předem připravených komponentů a zajišťuje rychlý a spolehlivý

běh aplikace. Většina frameworků poskytuje řešení pro často se objevující požadavky aplikace (4). V případě, že framework řešení neobsahuje, lze se obrátit na veliké množství knihoven a balíčků poskytované komunitou. Díky velké popularitě frameworků lze zaručit spolehlivý a otestovaný kód.

S rapidním nárůstem počtu frontendových frameworků může být výběr toho správného celkem složitý. Za zmínku stojí alespoň dva nejpopulárnější. Nejpopulárnějším frameworkem je aktuálně React. Byl vyvinut společností Facebook v roce 2013 a můžeme se s ním setkat na stránkách, jako je Netflix nebo Airbnb. Zajímavostí frameworku React je rozšíření syntaxe jazyka JavaScript JSX, který umožňuje propsání dat do komponentů. Největším konkurentem Reactu je Angular, framework vyvinutý společností Google v roce 2010. Angular na rozdíl od Reactu využívá programovací jazyk TypeSript[1], který rozšiřuje JavaSript o statické typování a další atributy známé z objektově orientovaného programování. Ovšem tato vlastnost může být pro některé vývojáře překážkou, pokud se s programovacím jazykem TypeScript doposud nesetkali (5).

3.4.1 React

Základním stavebním kamenem Reactu tvoří takzvané komponenty, což jsou znovupoužitelné HTML elementy se zapouzdřenou funkcionalitou. Komponenty pak mají vlastnosti (“props“) a spravují svůj vnitřní stav (“state“). Tento deklarativní způsob práce s daty aplikace vede k více předpověditelnému chování i lehčímu ladění (6).
React se využívá například k:

- Tvorbě tzv. single-page aplikací
- Vykreslování webových stránek na straně serveru (server-side rendering)
- Tvorbě nativních mobilních aplikací pomocí frameworku React Native

[1] Nadstavba nad jazykem JavaSript

3.5 Backend framework

Hlavní úlohou backendu je obsluha příchozích požadavků a komunikace s databází. V dnešní době je možné napsat backend v téměř jakémkoliv programovacím jazyce. Ale místo psaní backendu v oblíbeném programovacím jazyce je lepší podívat se na backend frameworky. Podobně jako u frontend frameworků se jedná o předem napsaný kód, který umožňuje jednodušší vývoj, správu a škálovatelnost. Poskytuje nástroje a knihovny pro zjednodušení vývoje obvyklých úloh, se kterými se může vývojář setkat. Backend frameworky dovedou nasměrovat požadavek na správné místo v kódu, komunikovat s databází nebo formátovat výstup. Využití backend frameworku dokáže vývojáři velmi ulehčit práci a zároveň zajistit bezpečný a spolehlivý kód.

Na jaké faktory se tedy zaměřit při výběru backend frameworku? Konkrétní požadavky na framework se mohou lišit na základě typu vyvíjené aplikace. Mezi hlavní faktory výběru patří rychlost frameworku nebo programovacího jazyka, v němž je framework napsán. Důležitým faktorem by měla být náročnost porozumění frameworku. Ideální je zaměřit se na frameworky, které jsou napsány v programovacím jazyce, který vývojář alespoň částečně ovládá. Dalším důležitým faktorem je kvalita dokumentace konkrétního frameworku. Během vývoje je běžné dohledávat potřebné informace v dokumentaci, její kvalita se tak váže na efektivitu práce. Za zmínku stojí také popularita frameworku. Pokud má framework dostatečně velkou komunitu, je vcelku jednoduché najít řešení k problémům, na které vývojář narazí během vytváření aplikace.

Mezi populární frameworky patří například Django napsaný v jazyce Python. Jedná se o rychlý, bezpečný a velmi škálovatelný framework, který používají stránky, jako je Instagram nebo Pinterest. Dalším frameworkem je Express vytvořený pro Node.js. Express je velice populární, především protože je napsán v jazyku JavaScript. Framework Express využívají například stránky Uber nebo IBM. Pro vývoj v typovaném jazyce, jako je Java, existuje například framework Spring Boot (7).

3.6 Databáze

Databáze je úložný systém, který shromažďuje organizovaná data. V kontextu webových aplikací se nejčastěji používají relační databáze, které jsou založeny na relačním modelu a využívají strukturovaný dotazovací jazyk SQL. Existují ale i databáze, které nepoužívají stejný způsob pro organizaci dat. Tyto databáze se nazývají nerelační databáze (8).

3.6.1 Relační databáze

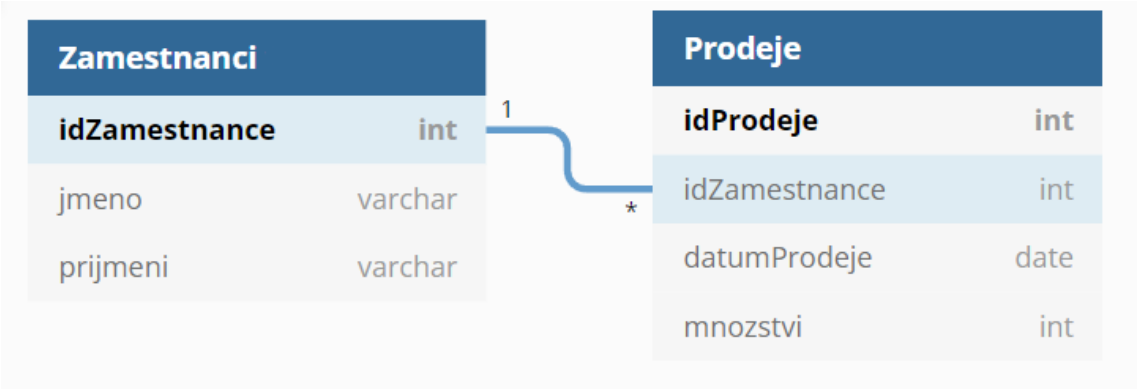
Relační databáze, také nazývané jako „Relational Database Management System“ (RDBMS) nebo SQL databáze, ukládají data do tabulek, jejichž řádky chápeme jako záznamy. Historicky nejpopulárnějšími relačními databázemi jsou Microsoft SQL Server, Oracle a MySQL.

Relační databáze dokážou propojit několik tabulek pomocí takzvaných „klíčů“. Klíč je v kontextu relačních databází unikátní identifikátor, který může být přiřazen k záznamu umístěném v tabulce. Tento unikátní identifikátor, označován jako „primární klíč“, může být také přidán do záznamu umístěném v jiné tabulce. Pokud je primární klíč přidán do záznamu jiné tabulky, je v rámci související tabulky označován jako „cizí klíč“. Spojení mezi primárním a cizím klíčem poté vytvoří relaci mezi záznamy umístěnými v několika tabulkách (9).

Následující obrázek demonstruje vazbu mezi dvěma tabulkami. Tabulka pojmenovaná Zamestnanci obsahuje řádky reprezentující jednotlivé zaměstnance, kterým byl přidělen unikátní identifikátor. V tomto případě je unikátní identifikátor pojmenován idZamestnance. Druhá tabulka, pojmenovaná Prodeje, obsahuje individuální záznamy o prodeji, které jsou vázány na konkrétního zaměstnance. Protože zaměstnanec dokáže realizovat několik prodejů, jeho unikátní identifikátor se může v tabulce Prodeje jako cizí klíč (9) objevit několikrát.

Jednou z velkých výhod relačních databází je takzvaná „referenční integrita“. Pravidlo referenční integrity vyžaduje, aby pro každý záznam v podřízené tabulce existoval záznam v tabulce nadřízené. Této integrity dat je dosaženo použitím

primárních a cizích klíčů. Referenční integrita by například zabránila odstranění zaměstnance z tabulky Zamestnanci, pokud by existoval záznam v tabulce Prodeje obsahující identifikátor zmíněného zaměstnance (9).



Obr. 2 Příklad fungování relačních databází
Zdroj: vlastní zpracování

Pro dotazování na data v relační databázi se používá strukturovaný dotazovací jazyk SQL. SQL umožňuje vytváření či získání záznamu, jejich aktualizaci a odstranění z tabulek.
Tento SQL dotaz demonstruje získání všech zaměstnanců z tabulky Zamestnanci, kteří se jmenují Jan.

```
SELECT * FROM Zamestnanci
WHERE Zamestnanci.jmeno = "Jan"
```

Obr. 3 Příklad SQL dotazu
Zdroj: vlastní zpracování

Relační databáze také poskytují funkci indexování. Databázový index je datová struktura, která zvyšuje rychlost získávání dat. Indexy jsou běžně přidávány na datová pole, která jsou často používaná při dotazování na data.

3.6.2 Nerelační databáze

Nerelační databáze, také označovány jako NoSQL databáze, na rozdíl od relačních databází neukládají data do tabulek. Místo tabulek používají datový model optimalizovaný pro specifické požadavky ukládaných dat. Mezi populární nerelační

databáze patří například MongoDB, Apache Cassandra nebo Redis. Nerelační databáze mohou například ukládat data v podobě dokumentů, „klíč-hodnota“ párů nebo grafů. Uložiště dokumentů spravuje sadu pojmenovaný polí a hodnot objektových dat v entitě označované jako „dokument“, uložené obvykle ve formátu JSON. Dokumenty mohou být ukládány v různých formátech, jako je XML, YAML, JSON nebo BSON. Uložiště dokumentů nevyžaduje, aby dokumenty udržovaly identické datové struktury, což vývojáři poskytuje velkou flexibilitu. Nerelační databáze díky absenci schémat umožňují rychlý vývoj a poskytují škálovatelnost a flexibilitu v případě změny požadavků (9).

4 Výběr a popis vybraných technologií

Pro vývoj funkční aplikace je zapotřebí výběr konkrétních technologií, které umožňují splnění všech požadavků. Ideálně by měly být zvolené technologie napsány v programovacím jazyce, se kterým je vývojář obeznámen, aby byl vývoj efektivnější. Důležitým aspektem technologie je také velikost komunity, ze které může vývojář čerpat informace.

4.1 Node.js

Node.js je multiplatformní, open-source runtime prostředí, které vývojářům umožňuje vytvářet aplikace v programovacím jazyce JavaScript. Umožňuje běh programů napsaných v jazyce JavaScript mimo prohlížeč. Node.js byl navržen se zaměřením na vysoký výkon a škálovatelnost webových aplikací a je dobrým řešením pro mnoho běžných problémů s vývojem webu. Pro vývojáře, kteří vyvíjí serverovou i klientskou část aplikace, je velikou výhodou použití stejného programovacího jazyka JavaScript. Node.js také poskytuje správce knihoven NPM, který umožňuje přístup k tisícům knihoven (10).

V rámci vyvíjené aplikace poskytuje Node.js prostředí pro běh backend frameworku Express, který obsluhuje HTTP požadavky, komunikuje s databází a sdílí soubory.

4.1.1 NPM

NPM neboli "Node.js package manager" je správce knihoven pro prostředí Node.js. NPM umožňuje vytvoření package.json souboru pomocí příkazu "npm init". Zmíněný soubor následně obsahuje informace o aplikaci. Mezi nejčastější parametry patří:

- Název
- Verze
- Skripty
- Odkaz na git repozitář
- Autor
- Seznam knihoven využívaných aplikací

Jedním z nejužitečnějších parametrů je bezprostředně seznam knihoven využívaných aplikací. Tento seznam se aktualizuje automaticky, a to vždy při instalaci nebo odinstalaci jednotlivých knihoven. Samotný seznam poté obsahuje název knihovny a její potřebnou verzi. Protože je zvykem instalované knihovny nezahrnovat do zdrojového kódu, NPM umožňuje jedním příkazem všechny potřebné knihovny nainstalovat. Takto například vypadá soubor package.json backednové části aplikace.

```
{
    "name": "websurvey-api",
    "type": "module",
    "version": "1.0.0",
    "description": "",
    "main": "app.js",
    "scripts": {
      "start": "node app.js"
    },
    "repository": {
      "type": "git",
      "url": "git+https://github.com/FruitSnack1/websurvey-api.git"
    },
    "author": "",
    "license": "ISC",
    "bugs": {
      "url": "https://github.com/FruitSnack1/websurvey-api/issues"
    },
    "homepage": "https://github.com/FruitSnack1/websurvey-api#readme",
    "dependencies": {
      "app-root-path": "^3.0.0",
      "body-parser": "^1.19.0",
      "cookie-parser": "^1.4.4",
      "cookies": "^0.8.0",
      "cors": "^2.8.5",
      "crypto": "^1.0.1",
      "crypto-js": "^4.0.0",
      "date-time": "^4.0.0",
      "dateformat": "^4.5.1",
      "dotenv": "^8.2.0",
      "excel4node": "^1.7.2",
      "express": "^4.17.1",
      "express-fileupload": "^1.1.7-alpha.4",
      "fs": "0.0.1-security",
      "geoip-lite": "^1.4.2",
      "imagemin": "^8.0.0",
```

```
    "imagemin-jpegtran": "^7.0.0",
    "imagemin-pngquant": "^9.0.2",
    "jsonwebtoken": "^8.5.1",
    "mongodb": "^3.5.2",
    "mongoose": "^5.8.11",
    "multer": "^1.4.2",
    "path": "^0.12.7",
    "qrcode": "^1.4.4",
    "rimraf": "^3.0.1"
  }
}
```

4.2 Express

Express patří mezi nejpopulárnější backend frameworky napsané pro Node.js. Umožňuje individuální zpracovávání požadavků s jinými URL cestami. Umožňuje vytváření takzvaných „middleware“ funkcí na libovolném místě v rámci zpracovávání požadavku. Přestože Express je sám o sobě vcelku minimalistický, existuje nespočet knihoven, které dokážou pracovat s cookies, URL parametry atd. V praxi Express funguje následovně. Čeká na požadavky zaslané klientem (v rámci vyvíjené aplikace je klientem frontend framework Angular). Po přijetí požadavku vykoná specifickou akci na základě podoby URL cesty. Může se jednat o získání či uložení dat do databáze, případně vykoná jiné úlohy pro správné obsloužení požadavku. Následně klientovi pošle odpověď v podobě dat nebo statusu, který upřesňuje, jak byl požadavek zpracován (10).

Následující příklad demonstruje minimum kódu potřebného pro vytvoření Express aplikace.

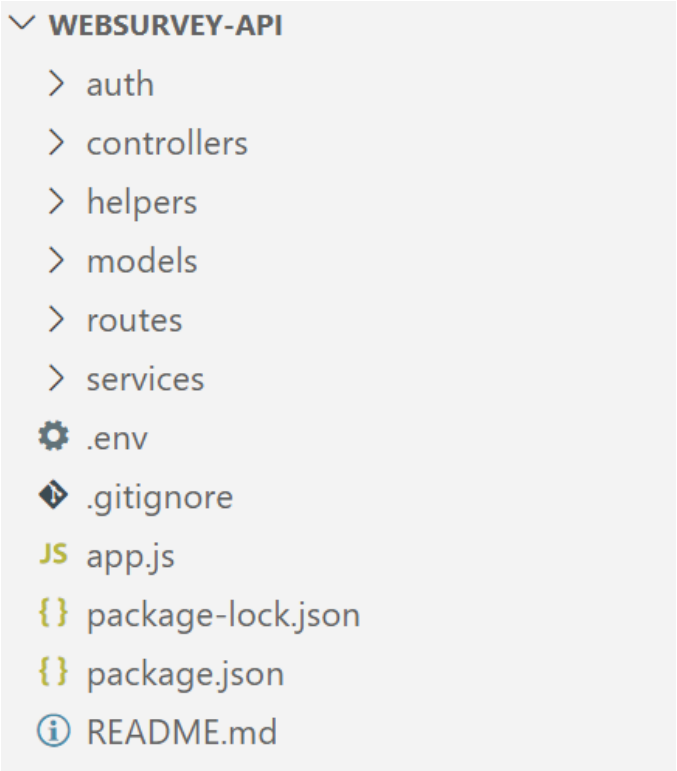
```
import express from 'express'
const app = express()

app.get('/rok', function (req, res) {
  res.send(`Aktuální rok je ${new Date().getFullYear()}`)
})

app.listen(3000)
```

Obr. 4 Příklad Express aplikace
Zdroj: vlastní zpracování

První dva řádky kódu obsahují import Expressu a následné vytvoření aplikace. Následuje funkce „get“, která obsahuje URL cestu a funkci, která zpracuje požadavek na zmíněnou URL cestu. Poslední řádek kódu spustí aplikaci na portu 3000. Pokud klient pošle požadavek na port 3000, obdrží jako návratovou hodnotu text obsahující aktuální rok.

4.2.1 Struktura projektu

Na rozdíl od jiných backendových framworků jako je například Spring Boot, Express nemá žádnou předem danou strukturu souborů. Tímto poskytuje vývojářům flexibilitu, ale při špatném použití může dojít k neuspořádanému kódu, který je špatně čitelný.



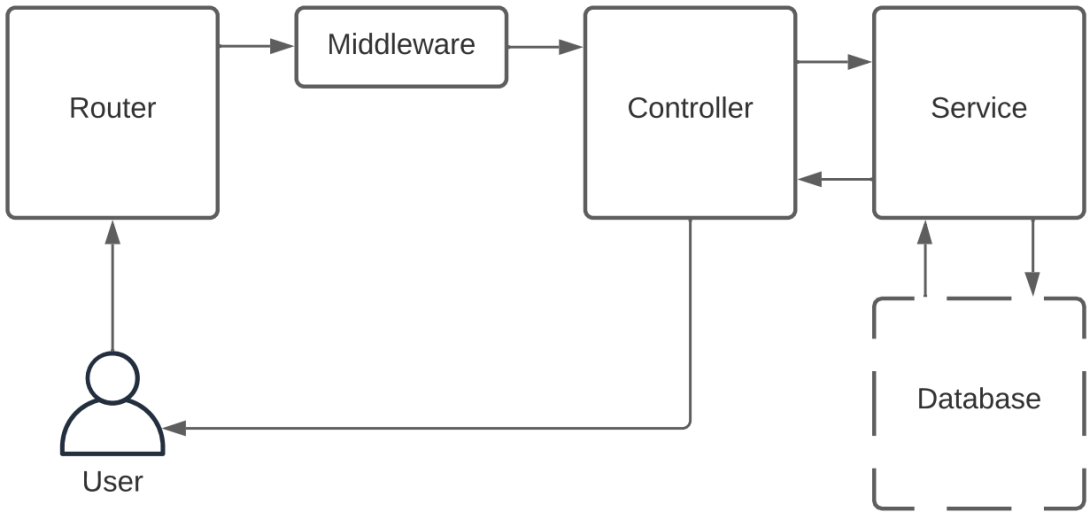
Obr. 5 Struktura backendu
Zdroj: vlastní zpracování

Ve vytvořené aplikaci byla použita tato struktura projektu, obsahující následující složky:

- **auth** – Obsahuje middleware funkce obsluhující autentifikační část aplikace.
- **controllers** – Obsahuje kontroléry, které jsou určeny k obsluze jednotlivých požadavků.

- **helpers** – Složka obsahující pomocné třídy. V případě vyvíjené aplikace obsahuje třídu, která umožňuje export výsledků do excelu.
- **models** – Obsahuje všechny potřebné datové modely.
- **routes** – Obsahuje jeden, nebo více souborů s logickým uspořádáním URL cest**.**
- **services** – Tato složka obsahuje třídy s business logikou aplikace. Soubory jsou děleny podle datových modelů a provádějí operace s databází.

V následujícím obrázku je zobrazen životní cyklus jednotlivých požadavků.



Obr. 6 Životní cyklus požadavku
Zdroj: vlastní zpracování

Požadavek přijde od klienta do Routeru, projde případnými middleware funkcemi a následně je přesměrován do předem definovaného Controlleru, který požadavek zpracovává dál. Controller poté volá funkce Service, která komunikuje s databází.

4.3 MongoDB

MongoDB je nerelační databáze, která ukládá data v podobě dokumentů. Je navržena pro ukládání velkého množství dat, se kterými umožňuje efektivně pracovat. Namísto tabulek a řádků, které používají tradiční relační databáze, MongoDB používá kolekce a dokumenty. Jednotlivé dokumenty obsahují páry „klíč-hodnota“ a kolekce pak obsahují samotné dokumenty. Každý dokument může mít jinou strukturu a může se lišit jak velikostí, tak obsahem. Protože se jedná o

nerelační databázi, není zapotřebí žádné předem definované schéma. Datový model v MongoDB vývojáři umožňuje reprezentovat hierarchickou strukturu pomocí ukládání dat do polí nebo objektů (11).

Tento příklad ukazuje, jakým způsobem mohou být data modelována v databázi MongoDB.

```
{
    "_id":"507f1f77bcf86cd799439011",
    "jmeno": "Jan",
    "prijmeni": "Novak",
    "prodeje": [
        {
            "idProdeje": "507f191e810c19729de860ea",
            "datumProdeje": "1639263600000",
            "mnozstvi": 22
        }
    ]
}
```

Obr. 7 Příklad dokumentu v databázi MongoDB
Zdroj: vlastní zpracování

Pole s názvem _id přiděluje dokumentům databáze automaticky. Jedná se o 24místný identifikátor přidělený ke každému dokumentu v databázi, který funguje jako primární klíč. Můžete si povšimnout, jak dokument obsahuje prodeje ve vnořeném dokumentu na rozdíl od relační databáze, kde by se prodeje ukládaly do samostatné tabulky viz obr. 1.

4.3.1 Mongoose

Absence schématu dat ukládaných do databáze poskytuje velikou flexibilitu. Umožňuje vývojáři kdykoliv upravit strukturu dat a vyhovět tak měnícím se požadavkům. Pokud ale chceme spolehlivou aplikaci, je potřeba data ukládaná do databáze nějakým způsobem validovat.

Mongoose je knihovna pro datové modelování v rámci MongoDB. Umožňuje vytváření specifických schémat v aplikační vrstvě. Zároveň poskytuje validaci a funkcionalitu zaměřenou na ulehčení práce s MongoDB. Pomocí knihovny Mongoose lze vytvořit schéma, které se váže na kolekci v databázi. Vytvořené schéma pak definuje specifickou strukturu dokumentů pro konkrétní kolekci. Ze zmíněného schématu se následně vytvoří model, který lze použít pro interakci s databází (12).

Tento příklad obsahuje schéma vytvořené pomocí knihovny Mongoose. Jedná se o schéma uživatele, které obsahuje uživatelské jméno, věk a e-mail. Jednotlivá pole mají předem definovaný datový typ a pole jméno a věk jsou pole povinná. Z tohoto schématu se následně vytvoří model, pomocí něhož se budou do databáze ukládat nové dokumenty. Před uložením knihovna Mongoose dokument validuje a uloží ho, pouze pokud splňuje veškeré podmínky definované ve schématu. Následně je dokument uložen do kolekce, na kterou je schéma vázané.

```
const schemaUzivatele = new mongoose.Schema(
  {
    jmeno: { type: String, required: true },
    vek: { type: Number, required: true },
    email: { type: String },
  },
  {
    collection: 'Uzivatele',
  }
)
```

Obr. 8 Příklad schématu knihovny Mongoose
Zdroj: vlastní zpracování

4.4 Angular

Angular je vývojářská platforma postavena na jazyce TypeScript obsahující:

- Framework pro tvorbu webových aplikací založený na komponentech
- Kolekci knihoven, které pokrývají širokou rozmanitost funkcionality
- Sadu vývojářských nástrojů pro vývoj, testování a kompilaci kódu

4.4.1 Komponenty

Komponenty jsou stavební bloky, které tvoří aplikaci. Každý komponent se skládá z TypeScript třídy, HTML šablony a CSS stylů.
Všechny komponenty obsahují:

- CSS selektor, který definuje, jak se se komponent používá v HTML šabloně (HTML elementy v šabloně se stejným selektorem se stanou instancí komponentu).
- HTML šablonu, která říká Angularu jak komponent vykreslit
- Nepovinný set CSS stylů, který definuje vzhled HTML elementů nacházejících se v šabloně

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Obr. 9 Příklad Angular kompnentu
Zdroj: (13)

4.4.2 HTML šablony

Každý komponent obsahuje HTML šablonu, která definuje, jak se má komponent vykreslit. Šablonu lze definovat přímo v kompnentu nebo v samostatném HTML souboru.
Angular rozšiřuje HTML extra syntaxí, která umožňuje vložení dynamických hodnot z komponentu. Při změně stavu komponenty pak Angular automaticky komponent překreslí.

```
<p>{{ message }}</p>
```

Obr. 10 Dynamické hodnoty v HTML šabloně
Zdroj: (13)

Angular také umožňuje dynamicky upravovat HTML strukturu pomocí podmíněného vykreslovaní nebo iterací nad listem hodnot.

4.4.3 Angular CLI

Angular CLI ("Command Line Interface") obsahuje sadu příkazů usnadňující vývoj. Příklady:

- **ng build** – Kompiluje Angular aplikaci
- **ng serve** – Vytvoří lokální webový server, který naslouchá změnám
- **ng generate** – Umožňuje generovat komponenty

4.4.4 Knihovny

Na rozdíl od Reactu, který pro často požadovanou funkcionalitu využívá knihovny třetí strany, Angular obsahuje svoje vlastní knihovny. Mezi nejpoužívanější patří například:

- **Angular Router** – Poskytuje kompletní řešení routování
- **Angular Forms** – Umožňuje vytváření komplexních formulářů a jejich validaci
- **Angular HttpClient** – Poskytuje klient-server komunikaci využívanou pro připojení k backend API

4.5 Hosting

V dnešní době existuje nespočet společností, které provozují hosting stránek. Záleží pouze na tom, jakou funkcionalitu vývojář vyžaduje. Společnosti nabízejí výhody jako servery s předinstalovanými aplikacemi, SSL certifikát nebo doménu v ceně.

4.5.1 Server

Pro hosting aplikace byl vybrán VPS (Virtual Private Server) od firmy Hetzner. Jedná se o již pronajatý server, který je k dispozici v rámci oddělení předvývoje. Server byl zabezpečen a chod aplikace byl zajištěn pomocí technologie Docker.

4.5.2 Doména

Doména je klíčovou součástí internetu. Poskytuje člověkem čitelnou adresu pro jakýkoliv webový server přístupný z internetu. Každý webový server připojený k internetu je přístupný pod veřejnou IP adresou. Tato IP adresa je ale pro člověka těžko zapamatovatelná a může se v budoucnu změnit. Doménu si ale nelze jen tak koupit. Lze si pouze koupit práva na její používaní, a to minimálně na dobu jednoho roku. Tento systém je zaveden proto, aby se nevyužité domény po určité době staly zase dostupné. Po pronajmutí domény lze změnit IP adresu, na kterou doména odkazuje (14).

V rámci aplikace byla využita společnost Wedos pro pronajmutí domény skodaquiz.com.

5 Vybrané aspekty implementace

5.1 Vývojové nástroje

V dnešní době existuje nespočet nástrojů pro vývoj softwaru. Umožňují editaci kódu, jeho testování, nebo opravování chyb.

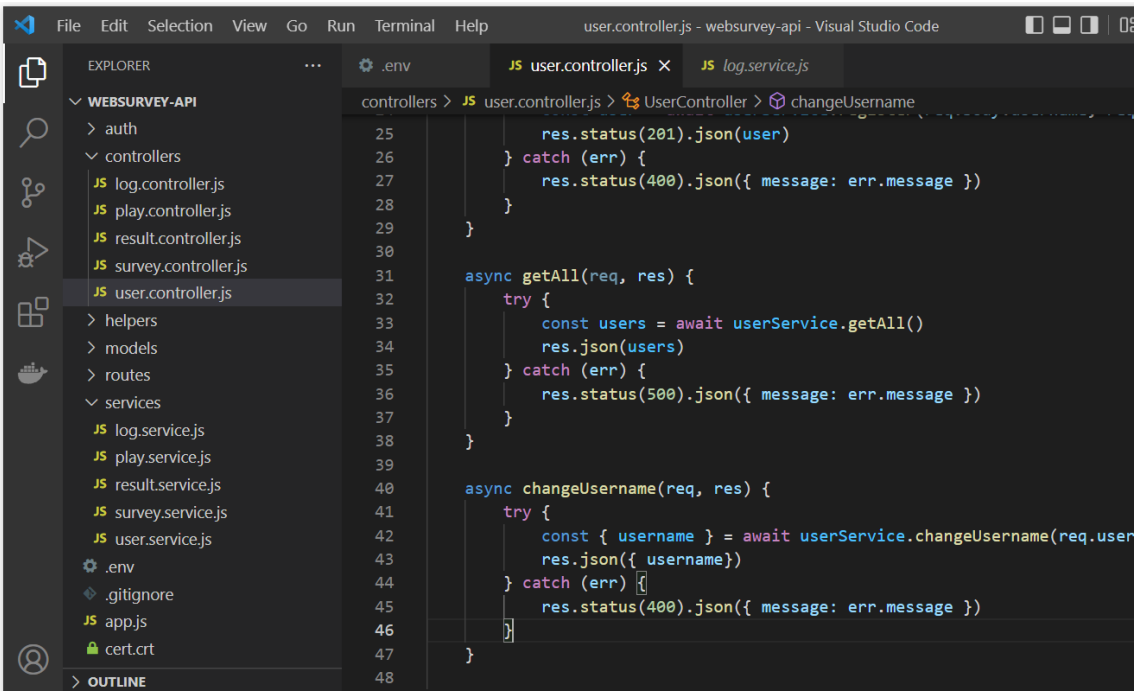
5.1.1 Editor

Jedním z nejdůležitějších nástrojů je editor. Na rozdíl od obyčejného textového editoru nabízí editor kódu zvýrazňování syntaxe, upozornění na chyby nebo kontextový našeptávač. Protože jakýkoliv editor umožňuje úpravu kódu je čistě na vývojáři, jaký editor se rozhodne použít.

Mezi nejpopulárnější patří bezprostředně editor Visual Studio Code. Jedná se o editor vyvíjený společností Microsoft pro operační systémy Windows, Linux a macOS. Mezi nejužitečnější funkce Visual Studio Code patří:

- Podpora pro Git
- Zvýraznění syntaxe
- Kontextový našeptávač
- Podpora pro ladění
- Integrovaný terminál

Za zmínku také stojí nespočet rozšíření editoru vyvíjených komunitou, které nabízí přidání programovacího jazyku, změnu vzhledu a další nástroje pro vývoj. Visual Studio Code umožňuje instalaci zmíněních rozšíření přímo z editoru.



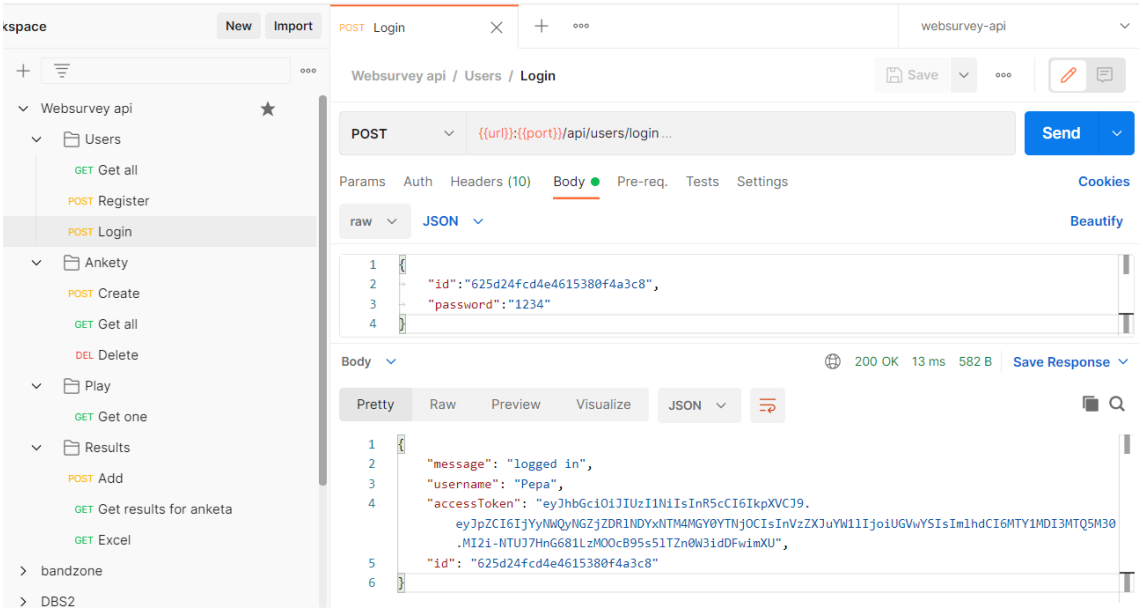
Obr. 11 Visual Studio Code
Zdroj: vlastní zpracování

5.1.2 Postman

Při vývoji API rozhraní je velmi důležitou částí jeho testování. Testování lze částečně provádět v prohlížeči, ale při používání komplikovaných požadavků je tato metoda nedostatečná. Mnohem lepším řešením je využití nástroje pro testování API rozhraní jako je Postman.

Postman nabízí (15):

- Vytváření kolekcí požadavků
- Editovatelné prostředí s vlastními proměnnými
- Editace hlaviček, parametrů nebo těla požadavku
- HTTP metody GET, POST, PUT atd.
- Webového klienta



Obr. 12 Postman
Zdroj: vlastní zpracování

5.1.3 Git

Git je distribuovaný systém pro správu verzí. Git sleduje veškeré změny provedené v souborech a vytváří o nich záznam, což umožňuje vývojáři návrat ke konkrétní verzi kódu podle potřeby. Největší výhoda Gitu nastává v momentě, kdy na kódu pracuje více lidí. Umožňuje více vývojářům provádět změny ve stejnou dobu a následné spojení změn do jednoho zdroje. Git umožňuje vytváření tzv. "repozitářů", které lze ukládat lokálně nebo online pomocí bezplatného softwaru jako je GitHub, GitLab atd. Samotný Git lze ovládat přes příkazový řádek nebo pomocí grafického rozhraní, které poskytuje například dříve zmiňovaný editor Visual Studio Code (16).

5.1.4 Vývojářské nástroje v prohlížeči

Jednou z nejdůležitějších věcí při vývoji klientské části aplikace je responzivita. Je velmi důležité, aby se webová stránka korektně zobrazovala na požadovaných typech zařízení. Na testování responzivity existují ve většině webových prohlížečů vývojářské nástroje. Kromě testování responzivity umožňují vývojářské nástroje také (17):

- Zobrazení a úpravu HTML elementů
- Zobrazení a úpravu kaskádových stylů

- Přehled chyb a upozornění ve vývojářské konzoli
- Testy výkonosti
- Sledování aktivity sítě

5.2 Zabezpečení dat

Jedním z prvních úkolů bylo zabezpečení aplikace. Konkrétně se jedná o vytvoření registrace a přihlášení, zabezpečení jednotlivých dotazníků a zabezpečení serveru, na kterém je aplikace spuštěná.

5.2.1 Přihlášení uživatele

HTML je nestavový protokol. To znamená, že server si nevede žádné záznamy o předchozích požadavcích klienta. V momentě, kdy chcete v aplikaci používat autentifikaci pro zabezpečení určitých zdrojů, je potřeba nějakým způsobem identifikovat uživatele, od kterého požadavky přicházejí. Technologie, pomocí které vyvíjená aplikace řeší tento problém, se nazývá JWT.

Zkratka JWT znamená JSON Web Token. Jedná se o internetový standard (RFC 7519), pomocí kterého lze bezpečně posílat informace v podobě JSON objektu. Samotný JWT se skládá ze tří částí. První částí je hlavička tokenu, která typicky obsahuje dvě informace, a to typ tokenu a typ algoritmu. Druhou částí tokenu je tělo, také nazýváno „payload“. Tělo tokenu obsahuje data o konkrétní entitě (typicky uživateli), kterou pomocí tokenu dokážeme identifikovat. Do těla lze uložit jakákoliv data ve formátu JSON, např. jméno, příjmení nebo uživatelskou roli. Poslední částí tokenu je digitální podpis. Pro vytvoření podpisu je potřeba zašifrovaná hlavička, zašifrované tělo, privátní klíč a algoritmus specifikovaný v hlavičce. Výstupem po podepsání je textový řetězec rozdělený tečkami, který lze jednoduše posílat v prostředí HTML (18).

V praxi JWT funguje následujícím způsobem. Uživatel serveru pošle přihlašovací údaje. Server ověří přihlašovací údaje a následně pomocí privátního klíče vygeneruje token obsahující informace o konkrétním uživateli. Server pošle uživateli nazpět zprávu o úspěšném přihlášení spolu s nově vygenerovaným

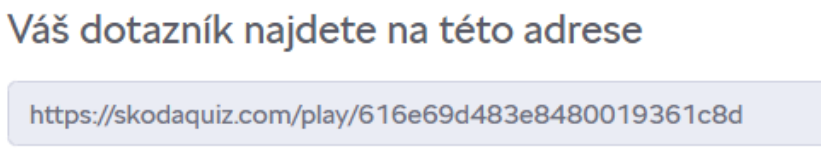
tokenem. Uživatel před odesláním jakéhokoliv následujícího požadavku přidá do hlavičky zmíněný token. Server pomocí privátního klíče token validuje, a ověří tak identitu uživatele.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Obr. 13 Podoba JSON Web Tokenu
Zdroj: (18)

5.2.2 Zabezpečení dotazníků

Vyplňování dotazníků obvykle patří mezi neoblíbené činnosti. Zatímco autorovi dotazníku dokáže poskytnout cenná data, pro jednotlivé respondenty se jedná o ztrátu času. Aby dotazník vyplnil co největší počet respondentů, je zapotřebí umožnit co nejjednodušší přístup. Zabezpečení dotazníku tedy bylo vyřešeno pouze vložením identifikátoru dotazníku do odkazu. I přesto, že byl dotazník veřejný, bez konkrétního odkazu je velmi náročné se k němu dostat.



Obr. 14 Podoba odkazu na dotazník
Zdroj: vlastní zpracování

5.2.3 Zabezpečení serveru

Jelikož byl firemní server již v minulosti napaden, bylo zapotřebí jeho zabezpečení. Ke vzdálenému serveru se připojuje pomocí služby SSH. Jednou z nejjednodušších forem zabezpečení je změna SSH portu. Změnou SSH portu lze zamezit nalezení serveru na portu 22, což je defaultní port služby SSH.

Nejčastějším útokem na server je takzvaný „brute force attack“. Jedná se o pokus o přihlášení do serveru využitím všech možných kombinací hesla. Tomuto útoku lze

zabránit instalací softwaru „fail2ban“. Tento software aktivně sleduje počet pokusů o přihlášení a při překročení předem definovaného limitu zablokuje IP adresu potencionálního útočníka.

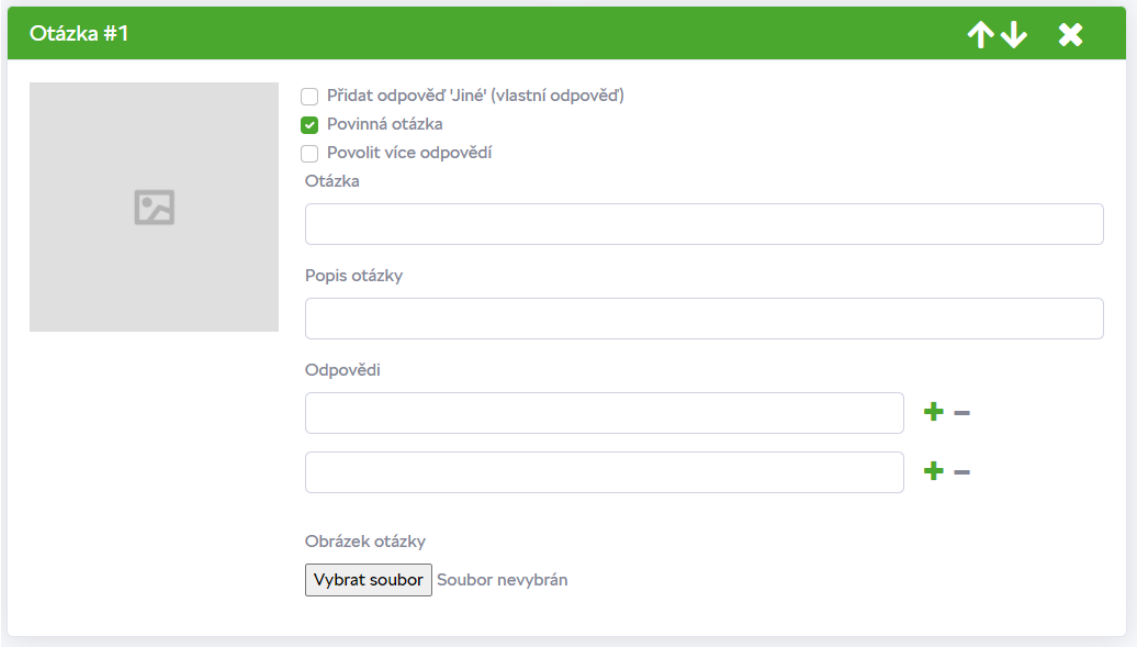
5.3 Tvorba dotazníků

Aby aplikace splnila veškeré požadavky, je zapotřebí vytvoření komplexních formulářů. Formuláře se nacházejí na klientské straně aplikace a budou tedy vytvořeny pomocí frameworku Angular. Na rozdíl od jiných frameworků nabízí Angular reaktivní formuláře umožňující vytváření dynamických formulářů a jejich validaci.

Reaktivní formulář je HTML formulář, který zpracovává svůj stav v podobě streamu dat v reálném čase (19). To znamená, že umožňuje naslouchat změnám a reagovat na tyto změny pomocí validace nebo databázových operací.
Mezi základní třídy reaktivních formulářů patří:

- **FormControl** – Individuální formulářové pole (textové pole, výběrové pole)
- **FormGroup** – Skupina formulářových polí, kterou lze validovat a manipulovat
- **FormBuilder** – Služby umožňující jednoduché vytváření reaktivních formulářů

Jednotlivé skupiny formulářových polí lze vnořovat do sebe a umožňují tak vytváření komplexních formulářů (19).



Obr. 15 Příklad formuláře v aplikaci
Zdroj: vlastní zpracování

5.4 Zpracování souborů

Jedním z požadavků na aplikaci byla možnost přidání obrázků k otázkám. Ukládání obrázku bylo vyřešeno pomocí ukládání do lokální složky na backendu. Důležitou vlastností ukládaných obrázků je jejich velikost. Nejenže soubory nadměrné velikosti zabírají místo na serveru, ale zároveň zpomalují načítání klientské části aplikace, což může znatelně ovlivnit uživatelskou zkušenost. Aby se zamezilo ukládání souborů nadměrné velikosti, byla použita knihovna “imagemin“ pro kompresi obrázků. Posledním úkolem bylo zpřístupnění obrázků uživatelům. Ve výchozím nastavení Expressu nejsou složky z webu přístupné. Pro zpřístupnění složky bylo tedy zapotřebí složku nastavit jako veřejnou, a to pomocí middleware funkce “express.static“.

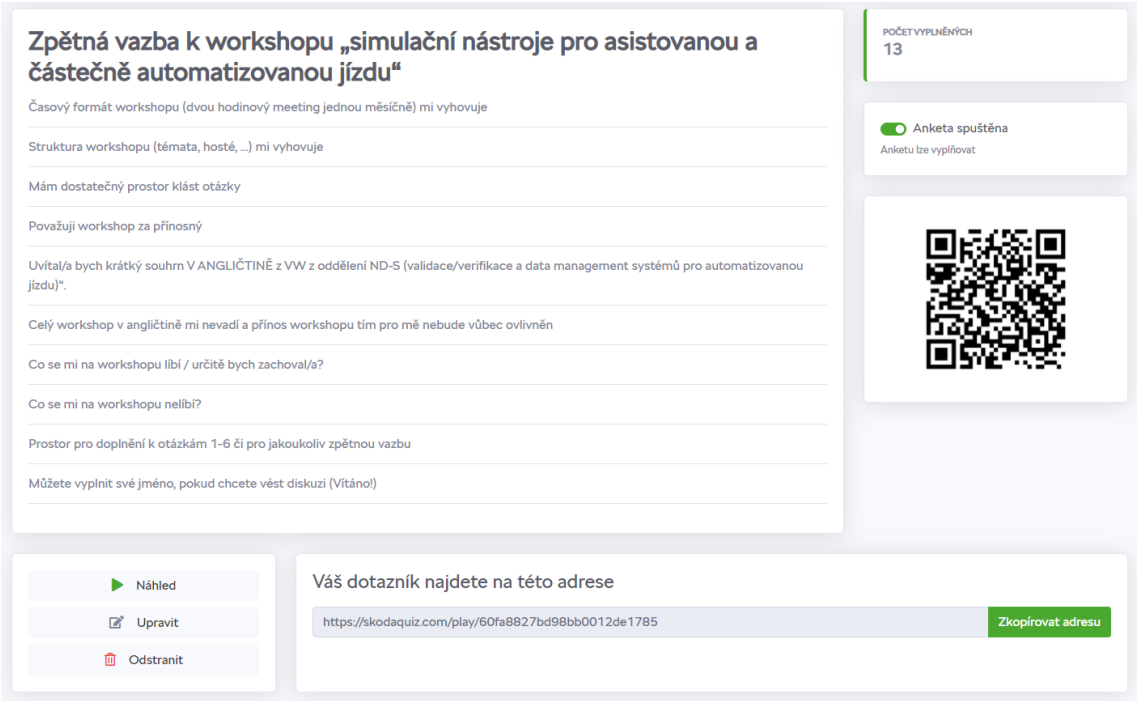
```
19    const app = express()
20
21    app.use(cookieParser());
22    app.use(bodyParser.json({ extended: false }))
23    app.use(express.static('public'))
```

Obr. 16 Použití middleware funkcí v Expressu
Zdroj: vlastní zpracování

5.5 Uživatelské prostředí

Podoba každého HTML dokumentu je definovaná pomocí jazyka CSS, také označovaného jako „kaskádové styly“. Může být vnořen do HTML dokumentu, nebo napsán v samostatném souboru. Jakou roli tedy mají kaskádové styly v rámci vyvíjené aplikace?

Aby byla vyvíjená aplikace vizuálně přívětivá, je zapotřebí zvolit vhodné kaskádové styly. Vždy existuje možnost napsat styly vlastní. V praxi se ale jedná o náročný a zdlouhavý proces, který často může vést k chybám v uživatelském prostředí. Jednodušší praktikou je použití CSS šablony, která obsahuje předem napsané kaskádové styly a umožní tak rychlý a efektivní vývoj aplikace.
Při vývoji aplikace byla využitá šablona CoolAdmin. Jedná se o open-source šablonu obsahující styly pro nejpoužívanější elementy, jako jsou tlačítka, karty, tabulky atd.



Obr. 17 Vizuál aplikace vytvořený pomocí CSS šablony
Zdroj: vlastní zpracování

5.6 Využití NFC na veletrhu inovací

Jelikož je zapotřebí nějakým způsobem sdílet odkazy na jednotlivé dotazníky s návštěvníky veletrhu, bylo zamýšleno použití QR kódů. Témata v rámci veletrhu

inovací ale bývají alespoň částečně utajená. Z tohoto důvodu je v celém prostoru inovačního veletrhu zakázáno pořizování fotografií. Proto bylo rozhodnuto o použití NFC tagů.

NFC čip je pasivní, znamená to, že nevyžaduje vlastní zdroj energie a je napájen pomocí elektromagnetické indukce. Po přiložení zařízení schopného čtení NFC čipů dochází k přenosu dat uložených na zmíněném čipu. Samotné čipy dokážou ukládat pouze velmi malé množství dat (20). V případě vyvíjené aplikace se jedná pouze o internetový odkaz.

Obr. 18 NFC tag
Zdroj: (20)

5.7 Další technologie využité při implementaci

Tato kapitola práce se zabývá dalšími technologiemi, které byly využity. Mezi využité technologie patří proxy server, reverse proxy server a Docker.

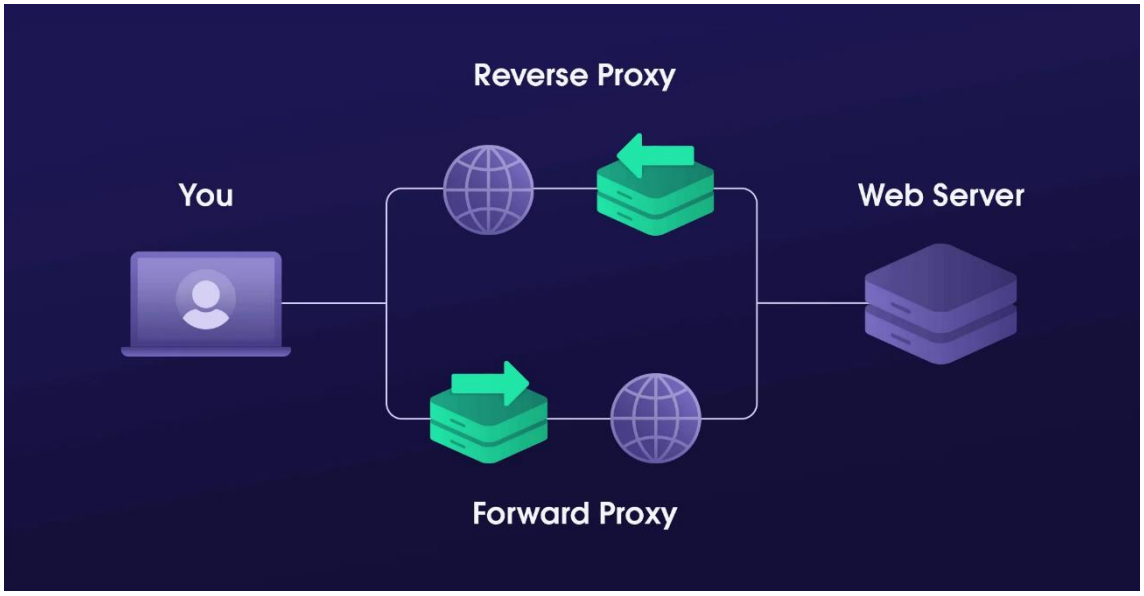
5.7.1 Proxy server

Proxy server funguje jako prostředník mezi uživatelem a internetem. Veškerá data z internetu tak projdou proxy serverem před tím, než dojdou k samotnému uživateli. Jedná se o standard u větších společností. Společnostem poskytuje extra vrstvu zabezpečení a nabízí možnosti, pomocí kterých lze kontrolovat nebo omezit aktivitu

zaměstnanců na internetu. Mnoho lidí také používá proxy server pro osobní účely (21).

5.7.2 Reverse proxy

Reverzní proxy server se na rozdíl od standardního proxy serveru nachází před serverem a přeposílá klientské požadavky na server. Používá se především pro lepší zabezpečení, vyšší rychlost a větší spolehlivost. Používá se také pro load balancing, kešování a SSL certifikáty. Za reverzním proxy serverem se může nacházet více serverů (22).



Obr. 19 Rozdíl mezi proxy servery
Zdroj: (22)

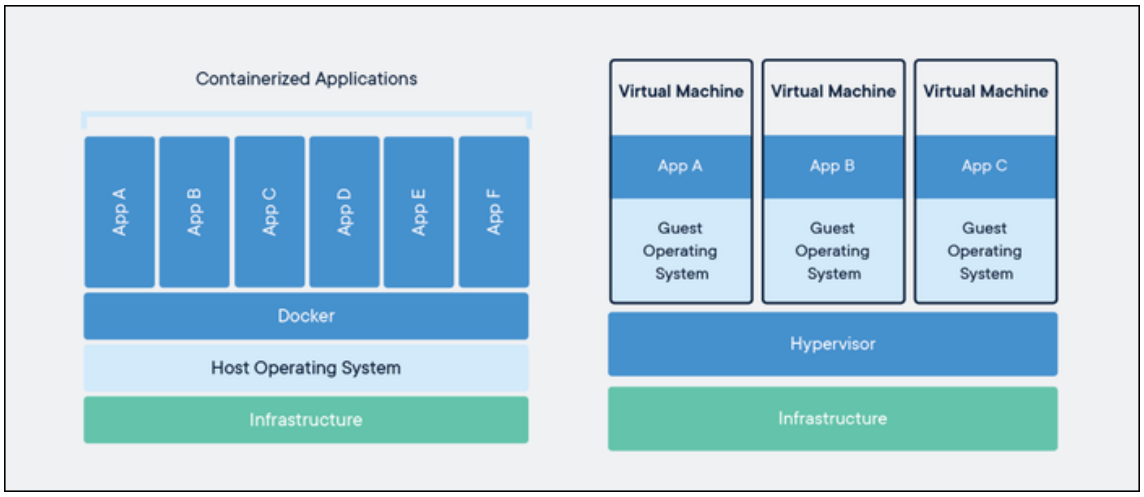
V rámci naší webové aplikace je využit reverzní proxy server pro dva účely. Prvním z nich je poskytování subdomén. Na pronajatém VPS jsou spuštěny celkem tři služby. Frontend, backend a databáze. Zatímco databázi nechceme zpřístupňovat uživatelům a přístup chceme ponechat pouze lokálně, chceme, aby ke zbylým službám uživatelé přístup měli. Pronajatý server ale poskytuje pouze jednu veřejnou IP adresu. Jakým způsobem lze tedy poskytovat přístup ke dvěma službám pod jednou IP adresou? Cílem je, aby uživatelé pod adresou https://skodaquiz.com našli frontend a pod adresou https://api.skodaquiz.com našli backend, který uživatelům poskytuje např. obrázky nebo excelové soubory. Reverzní proxy server se dokáže

podívat, z jaké URL adresy požadavek přišel, a následně požadavek přeposlat na server, který je na VPS spuštěn na libovolném portu.

Dalším účelem reverzního proxy serveru je SSL certifikát. Abychom nemuseli řešit SSL certifikáty na frontendu a zároveň na backendu, můžeme tuto akci provádět na jednom místě.

5.7.3 Docker

Docker je open-source projekt, který poskytuje jednotné rozhraní pro izolaci aplikací do kontejnerů. Jednotlivé kontejnery pak obsahují aplikační kód spolu s knihovnami operačního systému a veškerými požadavky pro běh zmíněného kódu v jakémkoliv prostředí. Docker nabízí sadu nástrojů pro vytváření kontejnerů, jejich nasazení a správu pomocí jednoduchých příkazů. Předchůdci Dockeru jsou klasické virtuální stroje, které na rozdíl od kontejnerů obsahují operační systém, a jsou proto pomalejší, výkonově náročnější a zabírají více místa. Docker umožňuje např. spuštění dvou Nginx serverů rozdílných verzí na jednom stroji (23).



Obr. 20 Rozdíl mezi Dockerem a virtuálním strojem
Zdroj: (24)

V kontextu vyvíjené aplikace jsou potřeba tři kontejnery. Kontejner pro frontend, backend a databázi. Jednotlivé kontejnery lze definovat pomocí souboru Dockerfile nebo stáhnout předem definovaný image z veřejného repozitáře Docker Hub. Pokud aplikace využívá více kontejnerů, lze využít souboru Docker Compose, pomocí

kterého lze specifikovat architekturu aplikace. Všechny kontejnery lze poté spustit pomocí jediného příkazu.

```
1   FROM node:14
2
3   WORKDIR .
4
5   COPY package*.json ./
6
7   RUN npm install
8
9   COPY . .
10
11  EXPOSE 3001
12
13  CMD ["npm", "start"]
```

Obr. 21 Dockerfile backendu
Zdroj: vlastní zpracování

Cílem souboru Dockerfile na Obr. 22 je vytvořit kontejner, ve kterém běží backendová Node.js aplikace. Příkaz "FROM" specifikuje image z kterého bude vytvořen image pro vyvíjenou aplikaci. V případě backendu se jedná o oficiální image Node.js verze 14. Docker následně zkopíruje soubor package.json, který obsahuje seznam všech knihoven na kterých je aplikace závislá. Zmíněné knihovny poté nainstaluje pomocí příkazu "npm install". Poté Docker zkopíruje zbytek souborů aplikace a zpřístupní port 3001. Nakonec Docker v kontejneru aplikaci spustí pomocí příkazu "npm start".

6 Výsledky

Aplikace byla otestována v rámci několika dotazníků vytvořených pro získání zpětné vazby. Kromě oddělení předvývoje využilo aplikaci také oddělení prodeje. Aplikace obstála zátěž uživatelů a během testů se nepřišlo na žádné závažné chyby, které by mohly negativně ovlivnit korektnost konečných výsledků či uživatelskou zkušenost respondentů.

6.1 Veletrh inovací

Největším využitím aplikace byl bezprostředně veletrh inovací pořádaný oddělením předvývoje. Po dobu čtyř dnů aplikace umožnila sběr zpětné vazby od návštěvníků inovačního veletrhu, a to bez jakýchkoliv problémů. Jelikož v minulosti veletrh inovací žádným způsobem zpětnou vazbu nesbíral, byly výsledky dotazníků velmi přínosné. A to jak pro oddělení předvývoje, tak pro samotné autory témat. Z 1157 návštěvníků veletrhu poskytlo zpětnou vazbu 26 %, přesněji 298 lidí. Návštěvníci vyplnili celkem 1784 dotazníků, a poskytli tak zpětnou vazbu 42 tématům prezentovaných na veletrhu.

Jelikož byly první tři otázky všech dotazníků totožné, bylo možné jednotlivá témata porovnávat mezi sebou. Počet výsledků byl ale převážně ovlivněn snahou prezentujících přesvědčit návštěvníky o vyplnění dotazníku.

7 Závěr

Cílem práce bylo vytvořit aplikaci pro tvorbu elektronických anket. Aplikaci se podařilo úspěšně vyvinout a zároveň splnit veškeré požadavky na funkcionalitu. Kromě samotného vývoje aplikace byla aplikace nasazena na zabezpečený soukromý server. Aplikace úspěšně odolala vysokému počtu uživatelů na veletrhu inovací, což je největší společenská akce, kterou oddělení předvývoje pořádá. Dokázala tak autorům témat poskytnout zpětnou vazbu. Aplikace je doposud využívána, a to jak k vytváření jednotlivých dotazníků, tak na každoročním veletrhu inovací.

Technologie vybrány pro vývoj aplikace byly dostačující. Umožnili realizaci vyžadované funkcionality a poskytli rychlý a stabilní běh aplikace. Zároveň také umožňují přidání dodatečné funkcionality v budoucnosti.

8 Seznam použité literatury

1. **York, John Flynn.** A Brief History of the Web. *Keepsite.* [Online] 22. 10 2015. [Citace: 18. 4 2022.] https://blog.keepsite.com/a-brief-history-of-the-web-809509ba23df.

2. **Senecki, Adrian.** Web development stacks – which stacks (should) we use in 2021? *The Software House.* [Online] 9. 7 2020. [Citace: 20. 1 2022.] https://tsh.io/blog/web-development-stacks/.

3. **Goins, Alexa.** Front End vs. Back End: What's the Difference? *Kenzie Academy.* [Online] 17. 8 2020. [Citace: 18. 1 2022.] https://kenzie.snhu.edu/blog/front-end-vs-back-end-whats-the-difference/.

4. **Patel, Tapan.** Does Your Web Application Need A Front-End Framework? *Third Rock Techkno.* [Online] 26. 3 2020. [Citace: 18. 1 2022.] https://www.thirdrocktechkno.com/blog/does-your-web-application-need-a-front-end-framework/.

5. **Hibbard, James.** The 5 Most Popular Front-end Frameworks Compared. *Sitepoint.* [Online] 13. 4 2021. [Citace: 20. 1 2022.] https://www.sitepoint.com/most-popular-frontend-frameworks-compared/.

6. **Máca, Jindřich.** Úvod do React. *ITnetwork.* [Online] [Citace: 17. 4 2022.] https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react/.

7. **Mozilla.** Database. *MDN Web Docs.* [Online] 17. 11 2021. [Citace: 20. 1 2022.] https://developer.mozilla.org/en-US/docs/Glossary/Database.

8. —. Server-side web frameworks. *MDN Web Docs.* [Online] 8. 10 2021. [Citace: 18. 1 2022.] https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks.

9. **Pattinson, Tamara.** Relational vs. non-relational databases. *Pluralsight.* [Online] 13. 8 2020. [Citace: 20. 1 2022.] https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases.

10. **Mozilla.** Express/Node introduction. *MDN Web Docs.* [Online] 18. 1 2022. [Citace: 21. 1 2022.] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

11. **Taylor, David.** What is MongoDB? Introduction, Architecture, Features & Example. *Guru99.* [Online] 1. 1 2022. [Citace: 20. 1 2022.] https://www.guru99.com/what-is-mongodb.html.

12. **Ado Kukic, Stanimira Vlaeva.** MongoDB & Mongoose: Compatibility and Comparison. *MongoDB.* [Online] 25. 11 2021. [Citace: 20. 1 2022.] https://www.mongodb.com/developer/article/mongoose-versus-nodejs-driver/.

13. **Angular.** What is Angular? *Angular.* [Online] 28. 10 2021. [Citace: 23. 1 2022.] https://angular.io/guide/what-is-angular.

14. **Mozzila.** What is a Domain Name? *MDN Web Docs.* [Online] 22. 1 2022. [Citace: 20. 1 2022.] https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_domain_name.

15. **Postman.** Tools. *Postman.* [Online] 2022. [Citace: 25. 4 2022.] https://www.postman.com/product/tools/.

16. **Noble Desktop.** What is Git and Why Should You Use it? *Noble Desktop.* [Online] 24. 10 2021. [Citace: 18. 4 2022.] https://www.nobledesktop.com/learn/git/what-is-git.

17. **Google.** Chrome Developers. *Overview.* [Online] 28. 3 2016. [Citace: 25. 4 2022.] https://developer.chrome.com/docs/devtools/overview/.

18. **JWT.** Introduction to JSON Web Tokens. *JWT.* [Online] [Citace: 21. 1 2022.] https://jwt.io/introduction.

19. **Delaney, Jeff.** Angular reactive forms basics guide. *Fireship.* [Online] 21. 3 2018. [Citace: 20. 2 2022.] https://fireship.io/lessons/basics-reactive-forms-in-angular/.

20. **Kilián, Karel.** Co je NFC a k čemu je dobré ho použít? *Svět Androida.* [Online] 29. 12 2021. [Citace: 20. 1 2022.] https://www.svetandroida.cz/co-je-nfc-k-cemu-je-dobre-ho-pouzit/.

21. **Petters, Jeff.** What is a Proxy Server and How Does it Work? *Varonis.* [Online] 21. 5 2018. [Citace: 21. 1 2022.] https://www.varonis.com/blog/what-is-a-proxy-server.

22. **Vistorskyte, Iveta.** Reverse Proxy vs. Forward Proxy: The Differences. *Oxylabs.* [Online] 8. 7 2021. [Citace: 21. 1 2022.] https://oxylabs.io/blog/reverse-proxy-vs-forward-proxy.

23. **IBM.** Docker. *IBM.* [Online] 23. 6 2021. [Citace: 21. 1 2022.] https://www.ibm.com/cloud/learn/docker.

24. **Heddings, Anthony.** What's the Difference Between Docker and a Virtual Machine (VM)? *CloudSavvy IT.* [Online] 7. 9 2020. [Citace: 21. 1 2022.] https://www.cloudsavvyit.com/5245/whats-the-difference-between-docker-and-a-virtual-machine-vm/.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Tomáš Rýzner**
Studium:	I1800219
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Systém pro realizaci elektronických anket**
Název bakalářské práce AJ:	System for realization of electronic surveys

Cíl, metody, literatura, předpoklady:

Cílem práce je vyvinout funkční řešení dotazníkové a hodnotící aplikace pro účely inovačních UX klinik
a interního inovačního veletrhu Škoda Auto.

Osnova:

1. Úvod
2. Sběr a analýza požadavků
3. Návrh řešení
4. Výběr a popis vybraných technologií
5. Vybrané aspekty implementace
6. Výsledky
7. Závěr

Zadávající pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	doc. Mgr. Tomáš Kozel, Ph.D.
Oponent:	Ing. Jaroslav Langer
Datum zadání závěrečné práce:	12.1.2021