Návrh a vývoj webovej aplikácie Handyman

Marek Mitrík

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Marek Mitrík**
Osobní číslo:	**A20105**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Návrh a vývoj webové aplikace Handyman**
Téma práce anglicky:	**Design and Development of a Handyman Web Application**

Zásady pro vypracování

1. Seznamte se s technologiemi a nástroji, které budou využity v praktické části, jako jsou HTML5, CSS3, JavaScript, React.js, Node.js a Figma.
2. Vytvořte analýzu trhu podobných webových aplikací zaměřených na nabídku a poptávku prací.
3. Na základě analýzy vytvořte funkční a nefunkční požadavky, které by měla webová aplikace Handyman splňovat. Pozornost věnujte i zabezpečení aplikace.
4. Prozkoumejte a porovnejte možná řešení tvorby aplikace Handyman v jiných nástrojích, jako jsou Angular nebo Vue.js.
5. Vytvořte návrh designu s využitím wireframe a UI prototypu.
6. Navrhněte a implementujte strukturu databáze a backend v rámci Node.js. Vytvořte API rozhraní pro propojení a vytvořte frontend aplikace pomocí React.js.

Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Slovenština**

Seznam doporučené literatury:

1. MINNICK, Chris. Beginning React JS Foundations building user interfaces with ReactJS. [Place of publication not identified]: John Wiley & Sons, 2022, 1 online resource. Dostupné z: doi:9781119685630
2. NGUYEN, Don. Node.js Okamžitě. Brno: Computer Press, 2016, 152 s. ISBN 9788025148204.
3. RICHARDSON, Leonard a Michael AMUNDSEN. RESTful Web APIs. Sebastopol: O'Reilly, 2013, xxviii, 373 s. ISBN 9781449358068.
4. NIEDERST ROBBINS, Jennifer, ed. Learning Web design: a beginner's guide to (X)HTML, style sheets, and web graphics. 3rd ed. Farnham: O'Reilly, 2007, 464 s. ISBN 9780596527525. Dostupné také z: https://digilib.k.utb.cz/handle/10563/52292
5. BOYD, Ryan. Getting started with OAuth 2.0. Sebastopol, CA: O'Reilly, c2012, x, 66 s. ISBN 9781449311605.

Vedoucí bakalářské práce: **Ing. Jakub Josef Forman**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25.5.2023

Marek Mitrík
podpis studenta

ABSTRAKT

Bakalárska práca je zameraná na vývoj webovej aplikácie Handyman, pomocou ktorého budú ľudia poskytovať a využívať ponúkané služby. Obsahom práce je vysvetlenie konceptov a terminológií o webovom vývoji, analýza trhu a porovnanie konkurenčných aplikácií, vytvorenie funkčných a nefunkčných požiadaviek, návrh dizajnu webovej aplikácie pomocou nástroja Figma a implementácia základnej funkcionality webovej aplikácie pomocou sady MERN stack.

Kľúčové slova: MERN stack, MongoDB, TypeScript, Express.js, Node.js, React.js, Figma, API

ABSTRACT

The bachelor thesis focuses on the development of a web application Handyman, through which people will provide and use the services offered. The content of the thesis is to explain the concepts and terminologies about web development, market analysis and comparison of competing applications, creating functional and non-functional requirements, designing the web application using Figma tool and implementing the basic functionality of the web application using MERN stack.

Keywords: MERN stack, MongoDB, TypeScript, Express.js, Node.js, React.js, Figma, API

Ďakujem môjmu vedúcemu bakalárskej práce, pánovi Ing. Jakubovi Josefovi Formanovi za odborné vedenie práce, vecné rady a možnosti diskusie ohľadne práce vždy, keď som potreboval. Tieto rady a pomoc boli kľúčovým prvkom k dokončeniu mojej bakalárskej práce. Taktiež Ďakujem všetkým, ktorí mi boli oporou a podporovali ma počas procesu písania bakalárskej práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V úvode tejto bakalárskej práce sa poskytuje podrobný pohľad na proces vývoja webovej aplikácie. Základné koncepty a terminológia budú predstavené s cieľom poskytnúť nevyhnutné vedomosti pre pochopenie a následnú implementáciu webových aplikácií.

Bude uskutočnená analýza trhu, kde sa detailne preskúmajú konkurenčné produkty a identifikujú sa príležitosti vznikajúce z ich nedostatkov pri vytváraní novej, zlepšenej platformy pre zákazníkov a poskytovateľov služieb.

Definovanie funkčných a nefunkčných požiadaviek bude kľúčovou súčasťou práce. Tieto požiadavky budú určovať očakávaný výkon a ciele webovej aplikácie. Na základe definovaných požiadaviek bude vytvorený návrh dizajnu webovej aplikácie, ktorý bude slúžiť ako vizuálny základ pre jej ďalší vývoj.

Počas procesu vývoja aplikácie bude kladený dôraz na implementáciu základných funkcionalít, vrátane vytvorenia nového účtu, prihlásenia, vytvorenia a prezerania zákaziek, filtrovania zákaziek, autorizácie a autentifikácie pomocou JWT, a tiež na zabezpečenie aplikácie proti potenciálnym kybernetickým hrozbám.

Cieľom práce je vytvorenie základnej verzie webovej aplikácie Handyman, ktorá by mohla slúžiť ako východiskový bod pre jej ďalší vývoj a potenciálne nasadenie do reálneho prostredia.

Táto bakalárska práca by mala poskytnúť jasný a detailný prehľad o procese vývoja webovej aplikácie, od konceptu až po implementáciu, a stať sa tak hodnotným zdrojom pre všetkých, ktorí majú záujem o túto oblasť.

I. TEORETICKÁ ČÁST

1 ROZBOR POUŽITÝCH TECHNOLÓGII

Táto časť slúži ako pomôcka pre bežných čitateľov, aby sa orientovali v praktickej časti. Na začiatku budú rozobrané základné webové technológie ako HTML, CSS, JavaScript. Následne sa ďalšie časti budú zaoberať Front-end technológiami ako React.js, Figma, Tailwind, ale aj back-end technológiami ako Node.js, Express.js, MongoDB, Mongoose.

1.1 HTML, CSS a JavaScript

HTML (HyperText Markup Language) je jazyk, ktorý sa používa na vytvorenie pevného základu toho, čo bude obsahovať daná webová stránka/aplikácia. Jedna sa o značkovací jazyk, ktorý využíva elementy na vytvorenie štruktúry webu od názvov, obrázkov, nadpisov až Hypertextových odkazov na presmerovanie na inú stránku. Príklad základného elementu je nasledovný:

Lorem Ipsum

Tento element sa skladá z nasledujúcich častí:

- otváracia značka
- Telo elementu Lorem Ipsum
- Uzatváracej značky

Každý element musí obsahovať tieto časti. Element taktiež môže obsahovať atribúty, ktoré obsahujú extra informácie o elemente. Tieto atribúty sa vždycky vpisujú do otváracej značky elementu. Dnešnej dobe sa využíva HTML verzia 5. [1]

CSS (Cascading Style Sheets) je jazyk, ktorý sa používa na dizajn webových stránok. CSS umožňuje meniť vzhľad webovej stránky, ako napríklad farby, fonty, veľkosti a pozície jednotlivých prvkov. Použitím CSS môžeme vytvoriť atraktívne a profesionálne vyzerajúce webové stránky bez toho, aby sme museli meniť samotný HTML kód stránky. [2]

JavaScript je programovací jazyk, ktorý zabezpečuje to aby webová aplikácia alebo stránka bola dynamická a interaktívna. V porovnaní s programovacími jazykmi ako je Java, C, C++ je JS interpretovaný jazyk, čo znamená, že jeho kód sa dokáže vykonávať priamo v užívateľovom prehliadači. Taktiež podporuje dynamické typovanie, ktoré zabezpečí to, že typy premenných sa pridávajú za chodu, takže premenné nepotrebujú preddefinovaný dátový typ.[3]

V dnešnej dobe je JS, najpopulárnejším programovacím jazykom na svete a to preto lebo je jednoduchý na naučenie sa, intuitívny a hlavne dokáže fungovať ako na klientskej strane tak aj na serverovej strane. Za jeho popularitu sa dá vďačiť aj tomu, že existujú desiatky frameworkov a knižníc, ktoré slúžia ako nadstavba pre bežný JS a ponúkajú developerom lepší „quality of life“ pri vývoji. Medzi najznámejšie JS frameworky a knižnice patrí: React.js, Angular, Vue.js a Node.js. [4][5]

HTML, CSS a JS sú pilierom každej modernej webovej aplikácie, kde HTML zabezpečuje obsah a štruktúru webu, CSS následne zabezpečuje dizajn a nakoniec JS zabezpečuje dynamickosť a interaktivitu danej webovej aplikácie.

1.2 TypeScript

TypeScript ďalej len TS je programovací jazyk, ktorý je nadstavbou JS, čo znamená, že akýkoľvek platným kódom v jazyku TS je platný aj v jazyku JS. TS však poskytuje ďalšie funkcie, ako je statické typovanie, rozhrania a triedy, ktoré z neho robia výkonnejší a škálovateľnejší jazyk ako JS. Medzi jeho prednosti patrí: typová ochrana, lepšia kvalita kódu atď. [6] Práve pre tieto dôvody by sa mal TS uprednostňovať pred JS pri vývoji.

1.3 React.js

React.js alebo React je JavaScript knižnica, slúžiaca pre vytváranie užívateľských rozhraní(UI) pomocou komponentov. Bol vytvorený spoločnosťou Meta v tom čase Facebook v roku 2011 pre svoje súkromne účely a však v roku 2013 prvá verzia Reactu bola vydaná pre verejnosť ako open source software. Dnes je používaný na mnohých populárnych webových stránkach a mobilných aplikáciách ako Facebook, Instagram, Netflix, Reddit, Airbnb atď. [7]

1.3.1 VIRTUAL DOM

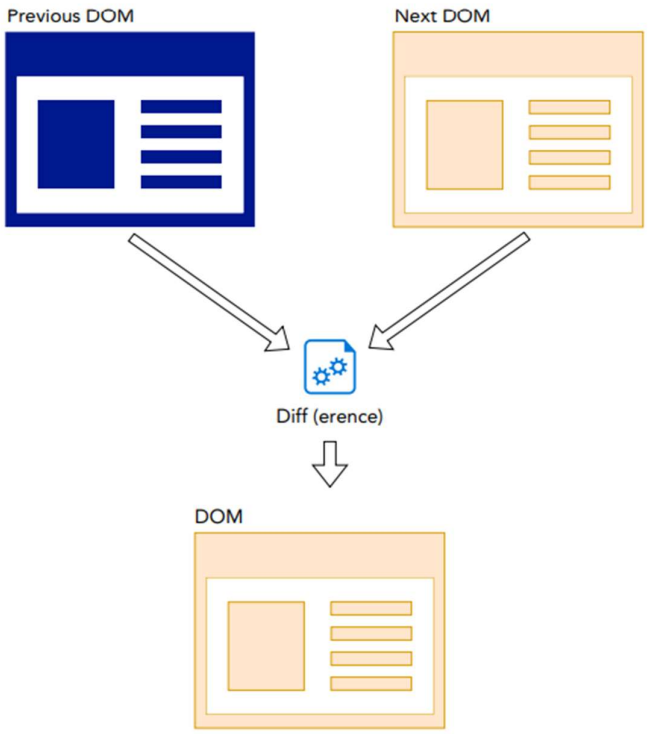
Objektový model dokumentu alebo DOM, je interná reprezentácia webovej stránky vo webovom prehliadači. Konvertuje HTML, štýly a obsah do uzlov, s ktorými možno pracovať pomocou jazyka JavaScript. Zmeny v DOM spôsobujú zmeny toho, čo sa dá vidieť vo webovom prehliadači a aktualizácie vykonané vo webovom prehliadači (napríklad pri zadávaní údajov do formulára) spôsobujú zmeny v DOM. Táto manipulácia s DOM je pomalá a neefektívna pretože vždy, keď sa DOM zmení, prehliadač musí skontrolovať, či si zmena bude vyžadovať stránku prekresliť a potom sa musí vykonať dané prekreslenie. [7]

React pristupuje k DOM pomocou Virtual DOM, ktorý slúži ako vrstva medzi kódom, ktorý píše programátor a DOM.

Tu je ukážka toho ako to funguje:

1. Programátor napíše kód v Reacte na vykreslenie používateľského rozhrania, ktorého výsledkom je jeden React prvok, ktorý je vrátený
2. Metóda vykresľovania ReactDOM vytvorí odľahčenú a zjednodušenú reprezentáciu React v prvku pamäti (Virtual DOM)
3. ReactDom počúva udalosti, ktoré si vyžadujú zmeny webovej stránky.
4. Metóda ReactDOM.render vytvorí novú reprezentáciu webovej stránky v pamäti.
5. Knižnica ReactDOM porovnáva novú virtuálnu reprezentáciu DOM webovej stránky s predchádzajúcou reprezentáciou Virtual DOM a vypočíta rozdiel medzi nimi. Tento proces sa nazýva zosúladenie.
6. ReactDOM aplikuje len minimálnu sadu zmien na DOM prehliadača z čo najefektívnejším spôsobom, ktorý dokáže a s použitím najefektívnejšieho dávkovania a časovania zmien

Tým, že programátor je vyradený z procesu skutočného vykonávania aktualizácií DOM prehliadača, ReactDOM môže rozhodnúť o optimálnom načasovaní a optimálnom spôsobe vykonávania požadovaných aktualizácií. Toto výrazne zvyšuje efektivitu vykonávania aktualizácií zobrazenia prehliadača. [7]



Obrázok 1. Ako funguje Virtuálny DOM

1.3.2 Rozmýšľanie v komponentoch

Rozmýšľanie v komponentoch je základným konceptom, keď sa pracuje v Reacte. Prinucuje to vývojárov na rozkúskovanie ich aplikácie na menšie, znova použiteľné častice, ktoré nazývame komponenty. Komponent môže byť napríklad navigačný panel, tlačidlo a ďalšie malé alebo veľké časti stránky. Tento modulárny prístup ma za následok to, že je jednoduchšie manažovanie, udržiavanie a škálovanie aplikácie. [8]

1.3.3 JSX

JSX alebo JavaScript XML je užitočný nástroj pre React developerov. JSX je rozšírenie jazyka JavaScript, ktoré poskytuje spôsob štruktúrovania vykresľovania komponentov pomocou syntaxe podobnej HTML. JSX ponúka možnosť zapisovať prvky HTML v jazyku JavaScript a umiestňovať ich do DOM tak, že konvertuje značky HTML na prvky React bez potreby ďalších metód, ako sú createElement() alebo appendChild(). Táto kombinácia JavaScriptu a HTML vedie k tomu, že aplikácie sú výkonnejšie. [9] [10]

1.3.4 Props

Props, skratka pre „properties“, sú v Reacte základným spôsobom, ako prenášať dáta a konfiguráciu medzi komponentami. Props umožňujú komunikáciu medzi nadradenými a podradenými komponentmi, čo zjednodušuje správu stavu a umožňuje vytvárať prispôsobiteľné a znovupoužiteľné komponenty. V Reacte sú props predávané z nadradeného komponentu do podradeného komponentu ako vstupné parametre. Podradený komponent potom používa props na čítanie dát alebo na konfigurácie od nadradeného komponentu. Dôležité je poznamenať, že props sú iba na čítanie, čo znamená, že ich hodnoty nemožno zmeniť priamo v podradenom komponente. [11] [12]

Príklad props je nasledovný:

```
const Hi = (name) => {
        return <h1>HI, {name} </h1>
}
function App(){
        return <Hi name=“Marek“/>
}
```

V tomto príklade nadradený komponent „App“ vytvára a zobrazuje podradený komponent „Hi“. Pri vytváraní komponentu „Hi“ nadradený komponent „App“ predáva prop „name“ s hodnotou „Marek“. Komponent „Hi“ potom používa túto hodnotu na vytvorenie pozdravu „HI, Marek“

1.3.5 Hooks

Hooks sú funkcie, ktoré sú súčasťou React knižnice od verzie 16.8 a umožňujú používať stav a ďalšie funkcie životného cyklu komponentu v rámci funkčných komponentov, bez potreby vytvárať triedy. Boli navrhnuté na riešenie niektorých problémov a obmedzení triedových komponentov, ako je napríklad zložitosť prenášania stavu a logiky medzi komponentmi, čo zjednodušuje a zefektívňuje vývoj a udržiavanie aplikácií v Reacte. [7] [13]

Medzi základné hooks patrí:

- useState: Slúži na manažovanie stavov v rámci komponentu. Je to jednoduchá funkcia, ktorá vráti dvojicu hodnôt: aktuálny stav a funkciu na jeho aktualizáciu [14]
- useEffect: Tento hook umožňuje vykonávať vedľajšie efekty, ako napríklad API volania, manipuláciu s DOM alebo aktualizáciu titulku stránky, v rámci funkčného komponentu. Je navrhnutý tak, aby nahrádzal metódy životného cyklu triedových komponentov, ako sú componentDidMount, componentDidUpdate a componentWillUnmount. [7]

Okrem týchto základných hookov, React ponúka hooky ako: useContext, useReducer, useRef, useCallback, useMemo atď. Taktiež developeri sú schopný vytvoriť si vlastné hooky na svoje unikátne prípady použitia ako napríklad data fetching, časovače atď. [13]

1.3.6 React Router

React používa na implementáciu smerovania v aplikáciách známy modul React Router. Bez toho, aby ste museli žiadať server o nové stránky, umožňuje spracovať dynamické zmeny v zobrazení aplikácie na základe adresy URL alebo histórie prehliadača. Výsledkom je zlepšenie používateľského zážitku a zvýšenie výkonu programu. [15]

1.4 Node.js

Node.js je open-source, multiplatformové prostredie a knižnica na spúšťanie webových aplikácií na servery. Bol vyvinutý v roku 2009 Ryanom Dahlom a postavený na JavaScriptovom engine V8 prehliadača Chrome, ktorý dovoľuje použitie vývojárom efektívne vyvíjať back-end, front-end a full-stack aplikácie pomocou JavaScript jazyka. Node.js používa asynchrónny beh riadený udalosťami, ktorý zabezpečuje vysokú škálovateľnosť a priepustnosť webových aplikácií, čo znamená, že umožňuje spracovávať viacero súbežných požiadaviek bez spomalenia systému. [16] [17]

1.4.1 Vlastnosti

Node.js má niekoľko kľúčových vlastností, ktoré odlišujú Node.js od ostatných back-end technológií a prispievajú k jeho popularite medzi vývojármi.

Vlastnosti sú nasledovné:

- Jednovláknový model: Node.js využíva jednovláknový model pre všetky požiadavky a sú zbierané v slučke udalostí. To znamená, že všetky programy sa vykonávajú v tom istom vlákne, od prijatia požiadavky cez dokončenie požadovanej úlohy až po odoslanie odpovede klientovi späť. Táto vlastnosť zabraňuje opätovnému načítaniu a redukuje čas prepínania obsahu, čo má za následok to, že Node.js je ekonomický na používanie [18]
- Node Package Manager(NPM): je vstavaný správca balíkov pre Node.js. Programátorom poskytuje prístup k rozsiahlemu úložisku balíkov, vďaka čomu je možné jednoducho zahrnúť a spravovať knižnice a závislosti v projekte Node.js. Okrem toho NPM podporuje správu verzií a zaručuje kompatibilitu. [19]
- Neblokovaný vstup/výstup: Neblokujúce I/O operácie sú jednou z hlavných predností Node.js. Pri bežnom synchrónnom spracovaní musí server počkať na dokončenie I/O operácie a až potom prejsť na ďalšiu požiadavku, napríklad na čítanie z databázy alebo súborového systému. Server môže stále spracovávať ďalšie požiadavky, pretože Node.js používa neblokujúcu, asynchrónnu techniku, pri ktorej sa I/O operácie vykonávajú na pozadí. Výsledkom je, že Node.js dokáže na jednom serveri spravovať tisíce spojení naraz. [18]
- Výkon: Node.js ponúka rýchle vykonávanie kódu, pretože je postavený na JavaScriptovom engine V8 od Google Chromu. Tento engine skompiluje JavaScript kód do strojového kódu, čo má za následok to, že kód je jednoduchšie a rýchlejšie implementovaný. Koncepty ako asynchrónne programovanie a spôsob, akým pracuje s neblokujúcimi vstupno-výstupnými operáciami, zabezpečujú jeho vysoký výkon. [18]

Tieto kľúčové vlastnosti pomohli Node.js získať obrovskú popularitu a stať sa štandardnou technológiou na vytváranie škálovateľných, efektívnych a vysoko výkonných webových aplikácií.

1.5 Express.js

Express.js alebo Express je najpopulárnejší back-end webový framework pre Node.js. Bol vytvorený na vývoj single-page, multi-page a hybridné webové aplikácie a takisto sa stal štandardom pre vývoj back-end aplikácií pomocou Node.js a to z dôvodu, že Express ponúka jednoduchosť, bez úkoru na jeho výkon a flexibilitu. [20] [21]

1.5.1 Middleware funkcie

Express operuje na základe sérií middleware funkcií. Middleware funkcie sú funkcie, ktoré majú prístup ku požiadavke objektu (request), odpovede objektu (response) a next funkcie v aplikačnom request-response cykly. Tieto funkcie môžu vykonávať akýkoľvek kód, modifikovať request a response ojekty, ukončiť request-response cyklus alebo zavolať ďalšiu middleware funkciu v poradí. Tento modulárny prístup dovoľuje vývojárom vysoký stupeň kontroly nad request-response cyklom. [20] [21]

1.5.2 Routing

Routing v Expresse je jedna z najsilnejších vlastností celého frameworku. Routing referuje ako majú aplikačné endopointy odpovedať klientovým požiadavkám. Express ponúka robustné riešenia pre routing, ktorý zahrňuje podporu pre URL parametre a zástupných znakov, čo uľahčuje vytváranie zložitých štruktúr API. [20] [21]

1.6 MongoDB

MongoDB je open-source NoSQL (Not only SQL) databázový systém, ktorý sa používa ako alternatíva k tradičným relačným databázam. Oproti relačným databázam MongoDB nevyužíva tabuľky a riadky na zapisovanie dát ale jej architektúra sa skladá z kolekcií a dokumentov. Kolekcia je ekvivalent SQL tabuľky, ktorá obsahuje súbory dokumentov. Dokumenty sa skladajú z key-value párov, čo je základná jednotka dát pre MongoDB. Tieto dokumenty používajú podobnú štruktúru ako JSON, ale sú nazývané Binary JSON (BSON). Výhodou tejto štruktúry je to, že pojme viacero dátových typov. Fieldy v dokumentoch sa chovajú podobne ako stĺpce v relačných databázach a ich hodnoty sa môžu skladať z rôznych dátových typov, môžu obsahovať ďalšie dokumenty, polia a polia dokumentov. [22]

Organizácie by mali uvažovať o použití MongoDB z nasledujúcich dôvodov:

- MongoDB môže ukladať veľké objemy štruktúrovaných a neštruktúrovaných dát a môže ich škálovať vertikálne a horizontálne. Indexy sú použité na vylepšenie výkonu pri vyhľadávaní. Vyhľadávanie sa vykonáva aj pomocou dopytov na polia, rozsahy a výrazy. [22]
- Databázy postavené na dokumentoch dovoľujú vkladať dokumenty na opis vnorených štruktúr a dokážu tolerovať odchýlky v údajoch [22]
- MongoDB môže byť použitá na chod cez viacero serverov [22]
- MongoDB používa replikáciu, kde súbor replík je sada dvoch alebo viacerých inštancií MongoDB, ktoré sa používajú na zabezpečenie vysokej dostupnosti. Tieto sady replík sa skladajú s primárneho a sekundárneho servera, kde primárny server vykonáva všetky operácie čítania a zápisu, zatiaľ čo sekundárna replika uchováva kópiu údajov. Ak primárna replika zlyhá, použije sa sekundárna replika. [22]

Navzdory tomu, aké benefity má MongoDB, taktiež má aj nedostatky ako:

- Kvôli automatickej stratégií „failover“ používateľ nastaví len jeden „Master node“ v klastri MongoDB. Ak tento uzol zlyhá, iný uzol sa automaticky stane novým „Master node“ a však tento proces môže trvať až minútu. [22]
- Jediný „Master node“ MongoDB taktiež limituje, ako rýchlo môžu byť dáta zapísané do databázy. Zapísané dáta musia byť nahrané na „Master node“ a zápis nových informácií do databázy je obmedzený kapacitou daného uzla. [22]
- MongoDB nedodáva úplnú referenčnú integritu pomocou obmedzení cudzích kľúčov, čo môže ovplyvniť konzistenciu údajov. [22]
- V databázach MongoDB nie je predvolene povolené overovanie používateľov. A však MongoDB pridalo predvolené nastavenie, ktoré blokuje sieťové pripojenia k databázam, ak neboli nakonfigurované správcom databázy [22]

1.7 Mongoose

Mongoose je dôležitý nástroj v MongoDB ekosystéme, vytvorený na to, aby zjednodušil proces pracovania s MongoDB v Node.js prostredí. Ako knižnica na modelovanie objektových údajov (ODM), zabezpečuje jednoduché riešenie založené na schéme pre modelova-

nie aplikačných dát, efektívne prepája medzeru medzi MongoDB bezschémovej nátury a potrebou spracovania štruktúrovaných údajov v logike aplikácie. [23]

1.7.1 Kľúčové vlastnosti Mongoose knižnice

- Schémy: Mongoose dovoľuje definovať objekty so „strongly-typed“ schémou, ktorá sa rovná kolekciám MongoDB. Tieto schému slúžia ako návod pre dokumenty MongoDB tak, že špecifikujú štruktúru údajov, predvolené hodnoty, validátori a iné, čím zavádzajú vrstvu štruktúry do inak flexibilného dátového modelu MongoDB. [24]
- Modely: Modely sú konštruktory vyššieho rádu postavené z definícií schém v Mongoose systéme. Reprezentujú MongoDB kolekcie a slúžia ako rozhranie pre CRUD operácie a dotazy. [24]
- Middleware: Knižnica Mongoose má podporu aj pre middleware, ktorý sa nazýva aj ako pre a post hooks a slúžia pre určité udalosti. Tieto middleware metódy, sa vykonávajú buď pred týmito udalosťami, alebo po nich a poskytujú účinný nástroj na konzistentnú správu údajov. [24]
- Vytváranie dotazov: Mongoose podporuje reťazové vytváranie dotazov, ktoré dopomáha vývojárom tak, že môžu písať MongoDB dotazy jednoduchšom a čitateľnejšom formáte. Tým pádom táto vlastnosť uľahčuje proces písania zložitých dotazov a zvyšuje čitateľnosť kódu. [24]
- Validácia: Mongoose poskytuje vstavanú validáciu pre schémy, dovoľujúca pre konzistentnú ochranu integrity dát. Taktiež Mongoose ponúka vývojárom vytvárať si vlastné validácie, ktorá ponúka flexibilné možnosti na dodržiavanie pravidiel pri dátach. [24]

Tieto vlastnosti robia z Mongoose go-to knižnicu pri práci s MongoDB.

1.8 MERN stack

MERN stack je JavaScript stack, ktorý bol navrhnutý tak aby zjednodušoval vývojárom proces vytvárania full-stack webových aplikácií. MERN stack je akronym, ktorý znamená M – MongoDB, E – Express.js, R - React.js a N – Node.js. [25]

MERN stack funguje nasledovným spôsobom:

- MongoDB slúži ako databázová vrstva aplikácie, ktorá ukladá dáta vo flexibilnom BSON formáte [25] [23]
- Express.js slúži ako serverová vrstva, ktorá beží na Node.js a pomocou neho sa vyvíjajú webové aplikácie, ktoré obsahujú sériu middleware volaní a routes. [25] [20]
- Node.js je technológia, ktorá spôsobuje, že JavaScript sa môže vykonávať aj na serverovej strane. [25] [16]
- React.js je front-end JavaScript knižnica, ktorá slúži na vytváranie užívateľských rozhraní, dovoľujúca efektívne aktualizovanie a renderovanie komponentov [25] [7]

Jeden z najzásadnejších výhod MERN stacku je ten, že celý tento stack je napísaný v jazyku JavaScipt, ktorý je momentálne najpopulárnejší programovací jazyk. To, že tento stack používa len jeden programovací jazyk spôsobuje to, že vývoj webovej aplikácie od front-end časti po back-end časti je veľmi príjemný zážitok pre vývojára pretože nemusí prepínať medzi rôznymi programovacími jazykmi a taktiež pre začínajúcich webových vývojárov je tento stack najlepšia voľba, kvôli jeho jednoduchosti a tomu, že je potrebný len jeden programovací jazyk aby sa dala vyvinúť plnohodnotná webová aplikácia. [25]

1.9 Figma

Figma je dizajnový nastroj založený na cloude, ktorý je známi kvôli jeho prívetivom rozhraní pre užívateľov a kolaboratívnym charakterom, vďaka čomu sa stala nástrojom pre jednotlivých dizajnérov aj veľké dizajnérske tímy. Používa sa pre vyhotovenie rôznych dizajnérskych úloh ako napríklad: dizajn užívateľského rozhrania, návrhu UX (user experience), prototypovanie, grafického dizajnu a mnoho ďalších. [26] [27]

Medzi základné funkcie patrí:

- Spolupráca v reálnom čase: Jednou z význačných vlastností Figmy, ktorá ju výrazne odlišuje a vyniká medzi ostatnými dizajnérskymi nástrojmi, je jej schopnosť umožniť spoluprácu v reálnom čase. Táto funkcia, ktorá bola v oblasti dizajnérskych nástrojov prvýkrát zavedená práve Figmou, umožňuje tímom spolupracovať na rovnakom dizajnovom súbore súčasne, pričom sa všetky zmeny automaticky aktualizujú. Táto možnosť výrazne zvyšuje efektivitu práce, umožňuje rýchlejšie ite-

rácie a zlepšenia dizajnu, čím sa stala jednou z kľúčových výhod Figmy pre kolektívnu a dynamickú tvorbu dizajnu. [26] [27]

- Prototypovanie: Figma dovoľuje dizajnérom tvoriť interaktívne prototypy bez zbytočných ťažkostí. Dizajnéri môžu spájať rôzne obrazovky, pridávať prechody, vytvárať mikroiterácie a tým Figma prispieva k tomu, aby sa mohli testovať používateľské zážitky bez potreby samostatného nástroja na prototypovanie. [26] [27]
- Komponenty a štýly: Figma podporuje znovupoužiteľné komponenty a štýly, čím uľahčuje udržiavanie konzistentného dizajnu v rámci systému návrhu v rámci projektu alebo celého tímu. Zmeny hlavných komponentov a štýlov sa premietnu všade, kde sa používaj, čím sa zabezpečí konzistentnosť a ušetrí čas. [26] [27]
- Založený na cloude: Tým, že Figma je cloudový nástroj, tak umožňuje dizajnérom pracovať odkiaľkoľvek bez toho, aby sa museli starať o synchronizáciu svojej práce. [26] [27]

Na základe týchto vlastností je Figma výborným nástrojom pre vzdialené tímy a rozsiahle projekty a taktiež je ideálnou voľbou pre návrh UI/UX. Figma je tiež multiplatformový nástroj, čo znamená, že ju možno využívať v akomkoľvek operačnom systéme, ktorý podporuje webový prehliadač. [26] [27]

Medzi možné alternatívy patria nástroje ako:

- Adobe XD: Adobe XD je nástroj na dizajnovanie a prototypovanie užívateľských rozhraní pre webové stránky a mobilné aplikácie. Ponúka funkcie ako je spolupráca v reálnom čase, integrácia s ostatnými Adobe produktmi a podpora pre rôzne platformy. [34]
- Sketch: Sketch je populárny nástroj na dizajnovanie užívateľských rozhraní, ktorý je k dispozícii iba pre macOS. Ponúka funkcie ako symboly a štýly, ktoré uľahčujú tvorbu konzistentných dizajnov. [35]
- Framer: Framer je nástroj na dizajn a prototypovanie, ktorý sa zameriava na tvorbu interaktívnych a animovaných dizajnov. Ponúka aj podporu pre React, čo umožňuje dizajnérom a vývojárom spolupracovať priamo na komponentoch aplikácie. [36]

II. PRAKTICKÁ ČÁST

2 ANALÝZA A NÁVRH WEBOVEJ APLIKÁCIE

Táto sekcia bakalárskej práce sa zaoberá analýzou a návrhom webovej aplikácie Handyman, ktorej cieľom je analyzovať trh s podobným produktom, vytvoriť funkcionálne a nefunkcionálne požiadavky na aplikáciu, ktoré by mali byť splnené pri finálnej verzii webovej aplikácie. Okrem toho je súčasťou návrhu aj vizualizácia webovej aplikácie pomocou dizajnového nástroja Figma.

2.1 Analýza trhu s podobným produktom

V tejto časti je realizované porovnanie webových aplikácií poskytujúcich služby analogické s tými, ktoré sú predmetom aplikácie Handyman. Špecificky sa venuje pozornosť stránkam 123dopyt.sk, aaadopyt.sk a poptávej.cz. Spoločným menovateľom týchto webových aplikácií je sprostredkovanie kontaktu medzi poskytovateľmi služieb a zákazníkmi hľadajúcich tieto služby. V nasledujúcich sekciách je rozobratá každá webová aplikácia, s dôrazom na služby, ktoré ponúkajú, hlavné funkcie a charakteristiky.

2.1.1 123dopyt.sk

Webová aplikácia 123dopyt.sk kategorizuje užívateľov do dvoch skupín: dopytujúci a dodávatelia. Dopytujúci majú možnosť zadarmo pridať dopyt, ktorý je následne zverejnený na webovej stránke a rozoslaný emailom relevantným dodávateľom, ktorí by mohli mať záujem skontaktovať dopytujúceho. Kontakt na dopytujúceho je následne poskytnutý potenciálnym dodávateľom pomocou emailu alebo telefónneho čísla. Registrácia nových dodávateľov na webovej stránke vyžaduje zadaní ich mena, priezviska a telefónneho čísla, na ktoré je následne zaslaný kontakt od zamestnanca webovej stránky 123dopyt. Dodávatelia sú tiež povinní platiť mesačný poplatok za používanie a ponúkanie svojich služieb, avšak na webovej stránke nie je uvedená konkrétna suma. Prvé dva mesiace používania sú pre dodávateľov zdarma, ale nie je jasne špecifikované, či je táto ponuka podmienená predplatným za dlhší časový úsek.

Pri prehľadávaní zadaných dopytov na stránke majú dodávatelia možnosť filtrovania podľa kategórií, lokality, hodnoty a dátumu zadania. Zoznam dopytov však nie je navrhnutý tak, aby umožňoval jednoduchú čitateľnosť. Dopyty sú umiestnené blízko vedľa seba a farba písma je dostatočne nápadná na to, aby pri čítaní jedného titulku upútala pozornosť na titulok iného dopytu. To môže znižovať celkovú prehľadnosť a efektivitu používania platformy pre dodávateľov.

Na základe tejto analýzy je možné identifikovať niekoľko oblastí, ktoré by mohli byť zlepšené v rámci webovej stránky 123dopyt.sk. Prvým bodom je absencia jasne uvedenej sumy mesačného poplatku pre dodávateľov, ktorá by mohla byť potenciálne odstrašujúca pre nové subjekty zvažujúce registráciu. Druhým bodom je celkový dizajn a usporiadanie zoznamu dopytov, ktoré by mohlo byť optimalizované pre lepšiu čitateľnosť a používateľskú skúsenosť. Tieto poznatky môžu byť využité pri návrhu a vývoji aplikácie Handyman, s cieľom vytvoriť produkt, ktorý je intuitívny a jednoduchý na použitie pre obe skupiny užívateľov - dopytujúcich aj dodávateľov.

2.1.2 Aaadopyt.sk

Aaadopyt.sk, podobne ako 123dopyt.sk, predstavuje platformu, ktorá sprostredkováva kontakt medzi dopytujúcimi a dodávateľmi. Funkčnosť oboch stránok je do veľkej miery porovnateľná, avšak existujú určité rozdiely, ktoré odlišujú Aaadopyt.sk.

Jedným z týchto rozdielov je dizajn a použiteľnosť formulára pre pridávanie dopytov. Na Aaadopyt.sk je tento formulár navrhnutý tak, aby bol pre užívateľa intuitívny a jednoduchý na použitie. To môže viesť k vyššej efektívnosti a spokojnosti užívateľa pri interakcii s webovou stránkou.

Rovnako ako na 123dopyt.sk, dodávatelia na Aaadopyt.sk sa musia zaregistrovať a zaplatiť predplatné za využívanie služieb. Cena za členstvo však nie je jasne uvedená, čo môže viesť k neistote a potenciálnemu odstrašeniu nových dodávateľov. Ponúka sa im dvojmesačné členstvo zdarma, avšak mechanizmus tohto systému nie je jasne vysvetlený, čo môže vytvárať ďalšiu nejasnosť a neistotu.

Pokiaľ ide o užívateľské rozhranie (UI), Aaadopyt.sk vykazuje vyššiu úroveň profesionality a estetického spracovania v porovnaní s 123dopyt.sk. Napriek tomu existujú oblasti, kde by sa mohla zlepšiť čitateľnosť a efektívnosť interakcie s užívateľom. Konkrétne je možné optimalizovať zoznam dopytov odstraňovaním nepotrebných prvkov, ako je profilová fotografia dopytujúceho, ktorá pre daný dopyt neprináša významnú hodnotu.

Na základe tejto analýzy je možné identifikovať dve hlavné oblasti pre zlepšenie. Prvou z nich je zlepšenie transparentnosti ohľadom cien členstva pre dodávateľov, čo by mohlo pomôcť eliminovať neistoty a zvýšiť dôveru užívateľov. Druhou oblasťou je zlepšenie užívateľského rozhrania, konkrétne optimalizácia zoznamu dopytov s cieľom zlepšiť čitateľ-

nosť a odstrániť nepotrebné elementy, ktoré môžu odvádzať pozornosť od podstatnejších informácií.

2.1.3 Poptávej.cz

Poptávej.cz predstavuje ďalšiu platformu, ktorá zohráva úlohu sprostredkovateľa medzi poskytovateľmi služieb a zákazníkmi, ktorí hľadajú tieto služby. Táto webová stránka, podobne ako predošlé, umožňuje klientom zadávať dopyty bezplatne. Rozlišujúca vlastnosť Poptávej.cz spočíva v transparentnosti jej cien pre členstvo.

Ďalším významným rozdielom je bezplatná a nezáväzná registrácia pre poskytovateľov služieb, čo znižuje bariéru vstupu na platformu. Táto funkcia môže vytvárať silnejší pocit dôvery a pohodlia pre poskytovateľov, ktorí sa zaujímajú o využitie platformy.

Poptávej.cz sa tiež odlišuje v tom, že umožňuje poskytovateľom služieb kontaktovať klientov priamo prostredníctvom stránky. Táto funkcia poskytuje vyššiu mieru interakcie a komunikácie medzi oboma stranami, čo môže viesť k efektívnejšiemu a rýchlejšiemu procesu zabezpečenia služieb.

Ďalším unikátnym prvkom je kreditový systém, ktorý umožňuje registrovaným firmám získavať kontaktné údaje klientov, najmä v stavebných oblastiach. Firmy majú možnosť dobíjať na svoj užívateľský účet určitú sumu kreditu, ktorý môžu využiť na zobrazenie kontaktov pri kreditových dopytoch z vybraných oblastí a lokalít. Tento systém pravdepodobne umožňuje firmám flexibilnejšie a cielenejšie využitie platformy.

Rozhranie Poptávej.cz je navrhnuté tak, aby bolo jednoduché a prístupné pre užívateľov. Celkový dizajn stránky a jej funkcie naznačujú, že tvorcovia sa snažili zabezpečiť, aby užívatelia mohli bez problémov prechádzať a využívať platformu.

2.1.4 Príležitosti pre vlastný návrh

Výsledkom analýzy trhu bola identifikácia hlavných konkurenčných webových aplikácií poskytujúcich podobné služby ako plánovaná aplikácia Handyman - 123dopyt.sk, aaadopyt.sk a poptávej.cz. Všetky tieto platformy zastupujú sprostredkovateľskú úlohu medzi dodávateľmi a dopytujúcimi stranami, ktoré hľadajú rôzne služby.

123dopyt.sk a aaadopyt.sk ponúkajú podobné modely, kde dodávatelia musia zaplatiť poplatok za členstvo, pričom ceny nie sú transparentné. Na druhej strane, poptávej.cz prináša

transparentnejší prístup s jasne uvedenými cenami a bezplatnou registráciou pre dodávateľov.

Rozhrania a užívateľské skúsenosti týchto webových stránok sa líšia, pričom každá stránka má svoje silné a slabé stránky. Bolo zistené, že niektoré prvky, ako je zoznam dopytov a kontaktné formuláre, by mohli byť navrhnuté lepšie, aby zlepšili UX.

Z analýzy je zrejmé, že existuje priestor na vytvorenie webovej aplikácie, ktorá by mohla vylepšiť slabé stránky súčasných riešení a ponúknuť užívateľom jednoduchší, transparentnejší a efektívnejší systém pre sprostredkovanie služieb. Budúca aplikácia Handyman by mohla využiť tieto príležitosti a ponúknuť unikátnu hodnotu pre svojich užívateľov.

2.2 Návrh webovej aplikácie

Na základe vykonanej analýzy trhu je možné formulovať návrh novej aplikácie, ktorá by zlepšovala existujúce riešenia a prinášala inováciu do súčasného trhu. Plánovaná aplikácia, nazývaná Handyman, sa zamýšľa využiť nový inovatívny prístup k komunikácii medzi dodávateľmi a dopytujúcimi.

Rozhodujúcou charakteristikou aplikácie Handyman bude centralizácia procesov. Všetky aspekty, vrátane komunikácie, platieb za zákazky, prijatia zákazky, otázok týkajúcich sa zákazky, výplaty osôb, ktoré dokončili zákazku, atď., budú spravované priamo cez aplikáciu Handyman. Toto riešenie zamedzí potrebe využívania aplikácií tretích strán a zvýši efektivitu a jednoduchosť procesov.

Ďalej, aplikácia Handyman nebude fungovať na princípe predplatného pre dodávateľov, ako je to bežné u súčasných riešení. Namiesto toho budú dodávatelia odmeňovaní po dokončení každej zákazky, pričom malá časť sumy bude pripísaná aplikácii ako poplatok za sprostredkovanie. Peniaze budú pripísané na účet dodávateľa v aplikácii, odkiaľ ich bude možné previesť na vlastný bankový účet.

Týmto spôsobom navrhovaná aplikácia Handyman ponúka inovatívne a užívateľsky príveticejšie riešenie, ktoré je zamerané na zlepšenie komunikácie a procesov medzi dodávateľmi a dopytujúcimi.

Je dôležité poznamenať, že popisovaný koncept aplikácie Handyman je v súčasnosti len v štádiu návrhu. Prezentovaná verzia v rámci tejto práce bude demo verzia, ktorá demonštruje základné funkcionality a pracovné procesy navrhovanej aplikácie. Komplexné implementácie, ako je plná integrácia platieb a pokročilé funkcie komunikácie, budú súčasťou

budúcej vývojovej cesty aplikácie. Toto demo má za cieľ ukázať potenciál navrhovanej aplikácie a jej schopnosť priniesť inovácie a zlepšenia do súčasného trhu.

2.3 Funkcionálne požiadavky a nefunkcionálne požiadavky

Na základe tvrdení, ktoré sú popísane v návrhu je ďalším krok vytvoriť funkcionálne a nefunkcionálne požiadavky, ktoré popíšu to čo musí obsahovať finálna verzia aplikácie Handyman.

2.3.1 Funkcionálne požiadavky

Funkcionálne požiadavky popisujú konkrétne funkcie alebo činnosti, ktoré systém alebo komponent systému musí byť schopný vykonať. Tieto požiadavky sa zaoberajú "čo" systém má robiť.

Funkcionálne požiadavky sú:

- Komunikácia: Aplikácia by mala umožniť bezproblémovú komunikáciu medzi dodávateľmi a dopytujúcimi.
- Správa zákaziek: Aplikácia by mala umožniť prijatie a dokončenie zákaziek prostredníctvom systému.
- Platby: Aplikácia by mala umožniť bezpečné spracovanie platby za zákazku a jej prenos na účet dodávateľa.
- Registrácia a prihlásenie: Aplikácia by mala poskytnúť možnosť registrácie a prihlásenia pre dodávateľov a dopytujúcich.
- Hľadanie a filtrovanie: Aplikácia by mala poskytnúť možnosti hľadania a filtrovania zákaziek podľa rôznych kritérií, ako sú lokalita, typ práce, hodnotenie dodávateľa a podobne.
- Hodnotenie a recenzie: Aplikácia by mala umožniť dopytujúcim hodnotiť a písať recenzie na dodávateľov po dokončení zákazky.
- Správa účtu: Aplikácia by mala poskytnúť možnosti pre správu účtu, ako sú zmena hesla, úprava osobných údajov, nastavenie platobných informácií a podobne.
- Notifikácie: Aplikácia by mala poskytnúť systém notifikácií, ktorý informuje užívateľov o dôležitých udalostiach, ako sú nové zákazky, správy od dodávateľov, platby a podobne.

- Súkromné správy: Aplikácia by mala umožniť bezpečnú komunikáciu medzi dopytujúcimi a dodávateľmi prostredníctvom súkromných správ.
- Zdieľanie zákaziek: Aplikácia by mala poskytnúť možnosť zdieľania zákaziek s inými užívateľmi alebo na sociálnych sieťach.
- Odoslanie a prijatie ponuky: Aplikácia by mala umožniť užívateľom odoslať ponuku na vykonanie zákazky a taktiež prijať alebo odmietnuť danú ponuku
- Diskusia pod zákazkou: Aplikácia by mala poskytovať funkcionalitu, ktorá umožňuje užívateľom vytvárať a sledovať diskusiu priamo pod detailmi zákazky. Tento dialóg by mohol slúžiť na vyjasnenie akýchkoľvek nejasností týkajúcich sa zákazky, poskytnutie dodatočných informácií alebo odpovedanie na otázky týkajúce sa zákazky. Diskusia by mala byť otvorená pre všetkých zainteresovaných strán s cieľom zabezpečiť plnú transparentnosť a porozumenie zákazky.

2.3.2 Nefunkcionálne požiadavky

Nefunkcionálne požiadavky sa týkajú "ako" systém vykonáva svoje funkcie, a zahŕňajú aspekty ako výkonnosť, bezpečnosť, spoľahlivosť, atď.

Nefunkcionálne požiadavky sú:

- Bezpečnosť: Aplikácia by mala implementovať štandardy pre bezpečnosť dát a ochranu osobných údajov, aby sa chránili informácie užívateľov. Mala by zahrňovať funkcie, ako sú silné heslá, šifrovanie dát a dvojfaktorové overovanie.
- Rozšíriteľnosť: Aplikácia by mala byť navrhnutá tak, aby bola ľahko rozšíriteľná a prispôsobiteľná pre budúce potreby a vývoj.
- Výkon: Aplikácia by mala byť rýchla a efektívna pri spracovaní požiadaviek užívateľov. Užívatelia by nemali čakať na načítanie stránky alebo spracovanie transakcií.
- Kompatibilita: Aplikácia by mala byť kompatibilná s rôznymi prehliadačmi a mobilnými platformami. Mala by sa prispôsobiť rôznym veľkostiam obrazoviek a rozlíšeniam.
- Používateľská prívetivosť: Aplikácia by mala mať intuitívne a jednoduché používateľské rozhranie. Užívatelia by mali byť schopní ľahko nájsť a využiť požadované funkcie bez potreby špeciálneho školenia alebo pomoci.

- Stabilita: Aplikácia by mala byť stabilná a bezchybná. Mala by byť odolná voči chybám a schopná sa rýchlo obnoviť po zlyhaní.
- Dokumentácia: Aplikácia by mala byť dobre dokumentovaná, aby užívatelia a vývojári pochopili, ako funguje a ako sa používa. Dokumentácia by mala zahŕňať užívateľské príručky, technické špecifikácie a API dokumenty.
- Jednoduché nasadenie a údržba: Aplikácia by mala byť navrhnutá tak, aby jej nasadenie a údržba boli jednoduché a efektívne. To zahŕňa schopnosť ľahko aktualizovať aplikáciu s minimálnym prerušením služby.
- Škálovateľnosť: Aplikácia by mala byť navrhnutá tak, aby bola schopná zvládnuť nárast počtu užívateľov a transakcií bez významného vplyvu na výkon alebo spoľahlivosť.
- Dostupnosť: Aplikácia by mala byť navrhnutá tak, aby bola dostupná pre užívateľov 24 hodín denne, 7 dní v týždni. To znamená, že musí byť navrhnutá tak, aby zvládla prerušenia služby a obnovila sa po nich.
- Zálohovanie a obnova dát: Aplikácia by mala obsahovať mechanizmy na zálohovanie a obnovu dát užívateľov, aby sa zabránilo stratám dát v prípade výpadku systému alebo inej katastrofy.

3 DIZAJN WEBOVEJ APLIKÁCIE

Dizajn webovej aplikácie Handyman bude realizovaný prostredníctvom nástroja na dizajn Figma. Tento nástroj umožňuje vytváranie komplexných a detailných prototypov, ktoré sú potom použité ako referencia pri implementácii front-endu.

V procese dizajnovania sa budú vytvárať jednotlivé komponenty, ktoré predstavujú stavebné bloky užívateľského rozhrania. Tieto komponenty budú potom použité na zostavenie rôznych obrazoviek aplikácie.

Dizajnované budú nasledujúce obrazovky:

- Domovská stránka: Predstavuje vstupný bod do aplikácie a poskytuje základné informácie o aplikácii.
- Prihlásenie: Stránka umožňujúca existujúcim užívateľom prihlásiť sa do aplikácie.
- Registrácia: Stránka poskytujúca možnosť novým užívateľom vytvoriť si účet v aplikácii.
- Pridanie novej zákazky: Stránka umožňujúca užívateľom vytvoriť a publikovať novú zákazku.
- Zobrazenie detailu zákazky: Stránka zobrazujúca všetky detaily konkrétnej zákazky, vrátane informácií o dopytujúcom užívateľovi, popisu práce, ceny a podobne.
- Zobrazenie zoznamu zákaziek: Stránka poskytujúca prehľad všetkých dostupných zákaziek, ktoré môžu byť filtrované a triedené podľa rôznych kritérií.
- Domovská stránka pre prihláseného užívateľa: Stránka, ktorá poskytuje personalizovaný prehľad aplikácie pre prihláseného užívateľa, vrátane aktuálnych zákaziek, noviniek a podobne.

Vytvorené dizajny budú slúžiť ako vizuálny návod počas implementácie front-endu webovej aplikácie.

3.1 Vytvorenie komponentov

Táto sekcia sa zaoberá vytvorením základných komponentov, ktoré budú základom pre dizajn webovej aplikácie Handyman. Tieto komponenty sú navrhnuté tak, aby boli konzistentné a univerzálne, čím sa zabezpečí jednotný a príjemný užívateľský zážitok. Komponenty sú nasledovné:

1. Výber farebnej palety: Farba je jedným z najdôležitejších prvkom v dizajne, pretože ovplyvňuje náladu a percepciu užívateľa. Pre aplikáciu Handyman bude vybraná farebná paleta, ktorá bude zahŕňať primárne, sekundárne a doplnkové farby.
2. Výber písma: Písmo je ďalším kľúčovým prvkom v dizajne, ktorý ovplyvňuje čitateľnosť a celkový vzhľad aplikácie. Pre aplikáciu Handyman bude vybrané písmo, ktoré je ľahko čitateľné a príjemné pre oči.
3. Vytvorenie loga: Logo je jedným z najdôležitejších prvkov značky. Je to prvý bod vizuálnej komunikácie, ktorý užívatelia identifikujú s Aplikáciou. Logo by malo obsahovať prvky, ktoré reprezentujú hodnoty a funkcie aplikácie a taktiež by malo byť relevantné k cieľovému publiku.
4. Vytvorenie navigačnej lišty (navbar): Navigačná lišta je dôležitým prvkom užívateľského rozhrania, ktorý umožňuje užívateľom ľahko prechádzať medzi rôznymi stránkami aplikácie. Budú vytvorené dva druhy navigačných líšt - jedna pre prihlásených užívateľov a druhá pre neprihlásených užívateľov.
5. Vytvorenie tlačidiel: Tlačidlá sú základnými interaktívnymi prvkami, ktoré umožňujú užívateľom vykonávať akcie, ako je prihlásenie, registrácia, odoslanie formulára a podobne. Pre aplikáciu Handyman budú vytvorené základné tlačidlá s konzistentným vzhľadom a funkčnosťou.

Z dôvodu časovej náročnosti a efektivity budú pre ďalšie potrebné komponenty použité preddefinované komponenty z dostupnej knižnice komponentov. Tieto komponenty budú prispôsobené tak, aby zodpovedali vizuálnemu jazyku a štýlu aplikácie Handyman.

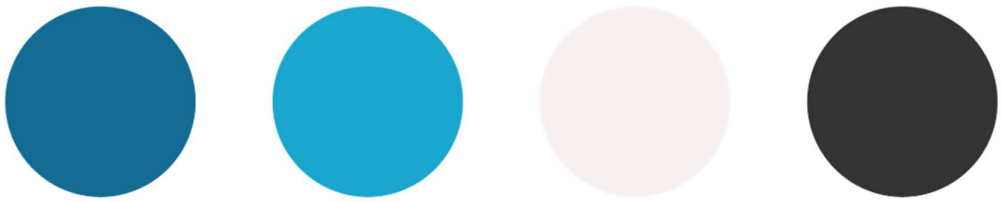
3.1.1 Výber farebnej palety

Farby sú kľúčovým prvkom v dizajne webových stránok a aplikácií, ktoré môžu na prvý pohľad vyvolať širokú škálu emócií a reakcií. Preto je dôležité vybrať farebnú paletu, ktorá vytvára príjemný a atraktívny dojem, zároveň podporuje odozvu a dôveru užívateľov. [28]

Webová aplikácia Handyman sa sústredí na poskytovanie spoľahlivých a bezpečných služieb, a preto je dôležité, aby farebná paleta odrážala tieto hodnoty. Modrá farba je často spájaná s dôverou, stabilitou a bezpečnosťou, a preto bola vybraná ako primárna farba aplikácie. Konkrétne, primárna (#146C94) bude dominovať dizajnu. [28]

Komplementárne farby sú zvolené tak, aby podporovali a dopĺňali primárnu farbu, čím umožňujú vytvorenie vizuálne príjemnej a harmonické farebnej schémy. Tieto farby zahŕ-

ňajú svetlejšiu modrú (#19A7CE), tmavú sivú (#33333) a svetlú sivú (#F6F1F1). Tieto farby dopĺňajú primárnu modrú a zároveň poskytujú dostatočný kontrast pre zvýšenie čitateľnosti a použiteľnosti aplikácie. [28]

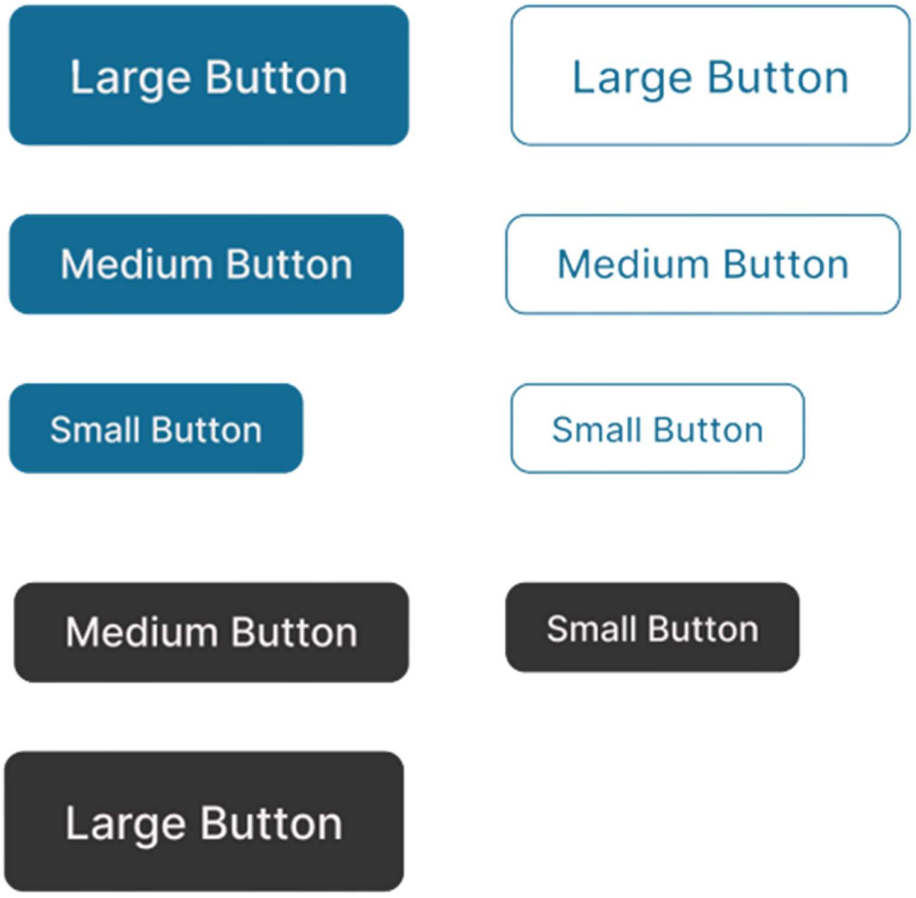


Obrázok 2. Farebná paleta

3.1.2 Vytvorenie tlačidiel

Tlačidlá sú zásadným interaktívnym prvkom každej webovej aplikácie, ktorý umožňuje užívateľom interagovať s rozhraním a navigovať cez rôzne funkcie a stránky. Významným faktorom v návrhu tlačidiel je ich viditeľnosť a čitateľnosť, pretože to priamo ovplyvňuje použiteľnosť a UX. [29]

V aplikácii Handyman budú tlačidlá navrhnuté tak, aby boli v súlade s celkovým dizajnom a farebnou paletou aplikácie. Bude k dispozícii až deväť variantov tlačidiel, ktoré sa budú líšiť vo farbách a veľkostiach. Tento výber umožňuje flexibilitu v návrhu rozhrania a zároveň zaistí, že tlačidlá budú v každej situácii jasne viditeľné a ľahko rozpoznateľné. Výber farieb a veľkostí bude v súlade s celkovým dizajnom a bude rešpektovať princípy použiteľnosti a prístupnosti. [29]



Obrázok 3. Tlačidlá

3.1.3 Výber písma

V procese výberu písma pre webovú aplikáciu Handyman bola kladená dôraz na čitateľnosť a estetiku. Bolo dôležité, aby bolo písmo dostatočne čisté a jednoduché pre ľahkú čitateľnosť na rôznych obrazovkách a aby malo moderný vzhľad, komplementárny k celkovému dizajnu aplikácie. [30]

Pre aplikáciu Handyman bol vybraný font Roboto. Roboto je často využívaný vo svete digitálneho dizajnu vďaka svojej jednoduchosti, čitateľnosti a pružnosti. Je charakterizovaný otvorenými tvarmi a plynulými krivkami, ktoré podporujú čitateľnosť textu pri rôznych veľkostiach a rozlíšeniach. Roboto ponúka aj širokú škálu hrúbok a štýlov, umožňujúc vytvorenie vizuálnej hierarchie a dôrazu v texte. [31]

Pre nadpisy a dôležité textové prvky bude použitý tučnejší variant Roboto fontu, zatiaľ čo pre bežný text a popisy bude uprednostnená jeho normálna a ľahká verzia. Toto riešenie umožňuje vytváranie konzistentného a harmonického vizuálneho jazyka v celej aplikácii.

Obrázok 4. Roboto font

3.1.4 Vytvorenie loga

Logo webovej aplikácie bolo navrhnuté s využitím nástroja Adobe Illustrator. Tvorba sa sústredila na integráciu elementov reprezentujúcich pracovné nástroje a samotný názov aplikácie. Výsledkom je jednoduché, ale efektívne logo, ktoré jasne vystihuje kľúčové vlastnosti a prednosti webovej aplikácie.

Obrázok 5. Handyman Logo

3.1.5 Vytvorenie navigačnej lišty (navbar)

Navigačná lišta bola navrhnutá s dôrazom na intuitívnosť a jednoduchosť používania. Jej cieľom je poskytnúť užívateľom jednoduchý a efektívny spôsob, ako sa pohybovať po webovej aplikácii Handyman.

Návrh navigačnej lišty zahŕňa dve verzie: jednu pre neprihlásených užívateľov a druhú pre prihlásených užívateľov.

Verzia pre neprihlásených užívateľov obsahuje možnosti pre prihlásenie a registráciu, ako aj odkazy na domovskú stránku a stránku s informáciami o službe.

Verzia pre prihlásených užívateľov ponúka prístup k užívateľovmu profilu, stránke so zákazkami, možnosti vytvoriť novú zákazku a odhlásiť sa.

V oboch verziách je logo aplikácie vždy viditeľné vľavo na navigačnej lište, aby poskytlo jasnú identifikáciu značky a umožnilo rýchly návrat na domovskú stránku.

Farba navigačnej lišty bola zvolená tak, aby zodpovedala farebnej palete aplikácie, pričom kontrast medzi textom a pozadím bol optimalizovaný pre čitateľnosť.

Obrázok 6. Navigačná lišta

3.2 Dizajn obrazoviek

V tejto sekcii je zameraná pozornosť na vytváranie dizajnu obrazoviek s využitím nástroja Figma. Tieto vizuálne návrhy budú slúžiť ako východiskový bod a referencia pre implementáciu front-endových častí práce. Dôležité je však zdôrazniť, že konečný dizajn a implementovaný výsledok nebude úplne identický, z dôvodu niekoľkých faktorov. Prvým z nich je časová náročnosť - dosiahnutie úplnej zhody medzi dizajnom a implementáciou by vyžadovalo výrazne viac času, ktorý je v rámci tohto projektu obmedzený. Druhým faktorom sú technické obmedzenia a kompromisy, ktoré sú často nevyhnutné pri prenose dizajnu do funkčnej aplikácie. Napriek tomu bude kladený dôraz na zachovanie kľúčových dizajnových prvkov a zabezpečenie vysokého štandardu použiteľnosti a vizuálnej príťažlivosti.

Navrhnuté budú nasledovné obrazovky:

- Domovská obrazovka: Táto obrazovka bude predstavovať vstupný bod do webovej aplikácie a bude obsahovať kľúčové informácie a navigačné prvky pre ďalšie časti stránky.
- Obrazovka na prihlásenie: Táto obrazovka bude navrhnutá tak, aby užívateľom umožnila prihlásiť sa do svojho účtu v aplikácii.
- Obrazovka na registráciu: Táto obrazovka bude slúžiť na vytvorenie nového účtu v aplikácii a bude obsahovať formulár pre zadanie potrebných údajov.
- Obrazovka na pridanie novej zákazky: Táto obrazovka bude obsahovať formulár pre vytvorenie a odoslanie novej zákazky.
- Obrazovka na zobrazenie detailu zákazky: Táto obrazovka bude poskytovať podrobné informácie o konkrétnej zákazke.
- Obrazovka pre zoznam zákaziek: Táto obrazovka bude poskytovať prehľad všetkých dostupných zákaziek v aplikácii.

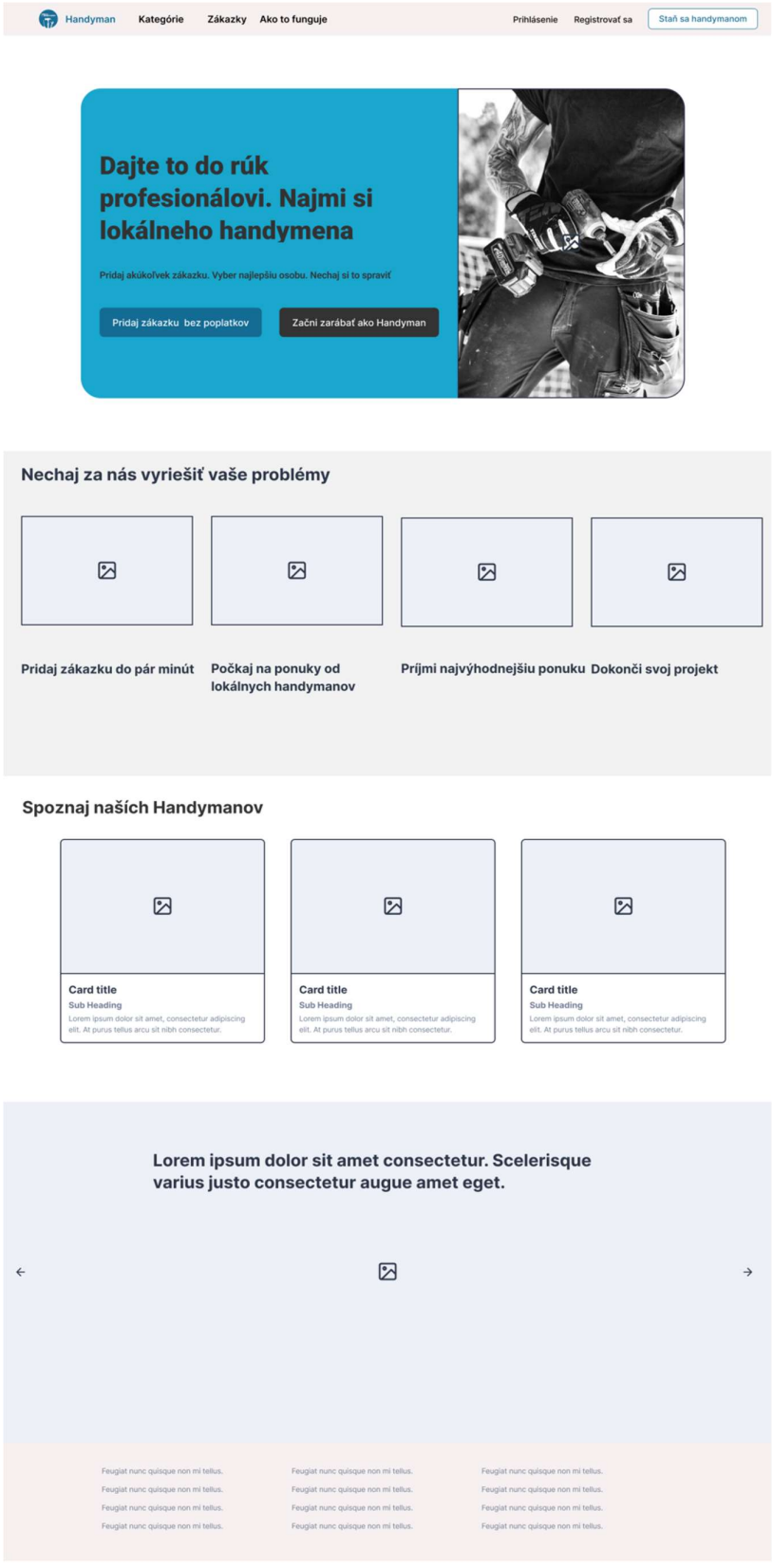
Každá z týchto obrazoviek bude navrhnutá s ohľadom na použiteľnosť a estetický vzhľad, pričom budú dodržané konzistentné dizajnové princípy a štýly stanovené v rámci vytvorených komponentov.

3.2.1 Domovská obrazovka

Domovská obrazovka bude slúžiť ako vstupný bod do webovej aplikácie, pričom jej dizajn bude zameraný na prilákanie a angažovanie potenciálnych zákazníkov. Bude sa skladať z nasledujúcich sekcií:

1. Hero sekcia: Táto sekcia bude umiestnená na vrchu obrazovky a bude obsahovať dve tlačidlá. Prvé tlačidlo bude určené pre užívateľov, ktorí chcú ponúkať svoje služby, a druhé pre užívateľov, ktorí chcú vytvoriť novú zákazku. Okrem toho bude táto sekcia obsahovať text, ktorý má za úlohu osloviť potenciálnych zákazníkov, a relevantný obrázok.
2. Sekcia s vysvetlením procesu: Táto sekcia bude nasledovať hneď po hero sekcii a jej úlohou bude oboznámiť užívateľa s procesom vytvorenia a dokončenia zákazky.
3. Sekcia s predstavením handymanov: V tejto sekcii budú predstavení úspešní handymani registrovaní v aplikácii, ktorí ponúkajú svoje služby. Cieľom tejto sekcie je zoznámiť potenciálneho zákazníka s rôznymi typmi služieb dostupných v aplikácii.
4. Carousel s recenziami: Posledná sekcia bude obsahovať carousel, ktorý bude zobrazovať recenzie od užívateľov. Tento prvok pomáha budovať dôveru a zabezpečiť transparentnosť služby.

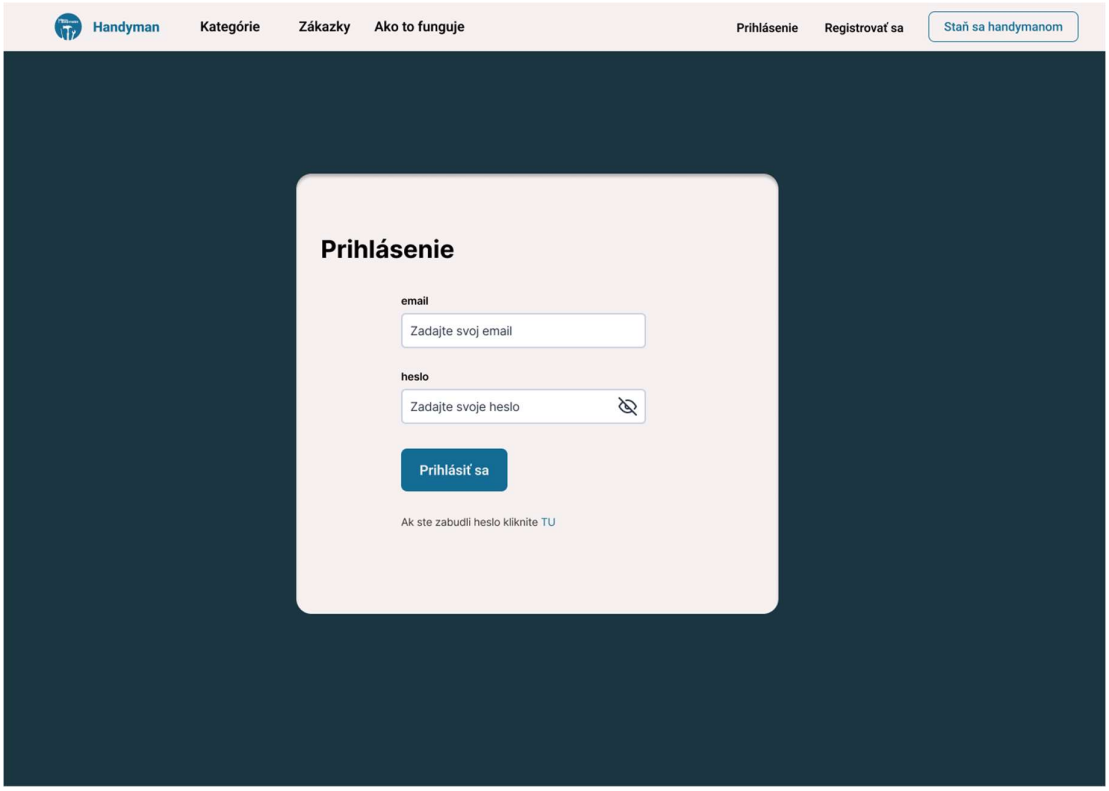
Celkový dizajn domovskej obrazovky bude usporiadaný tak, aby poskytol užívateľom jednoduchý a intuitívny spôsob prechádzania a interakcie s obsahom.



Obrázok 7. Domovská obrazovka

3.2.2 Obrazovka na prihlásenie

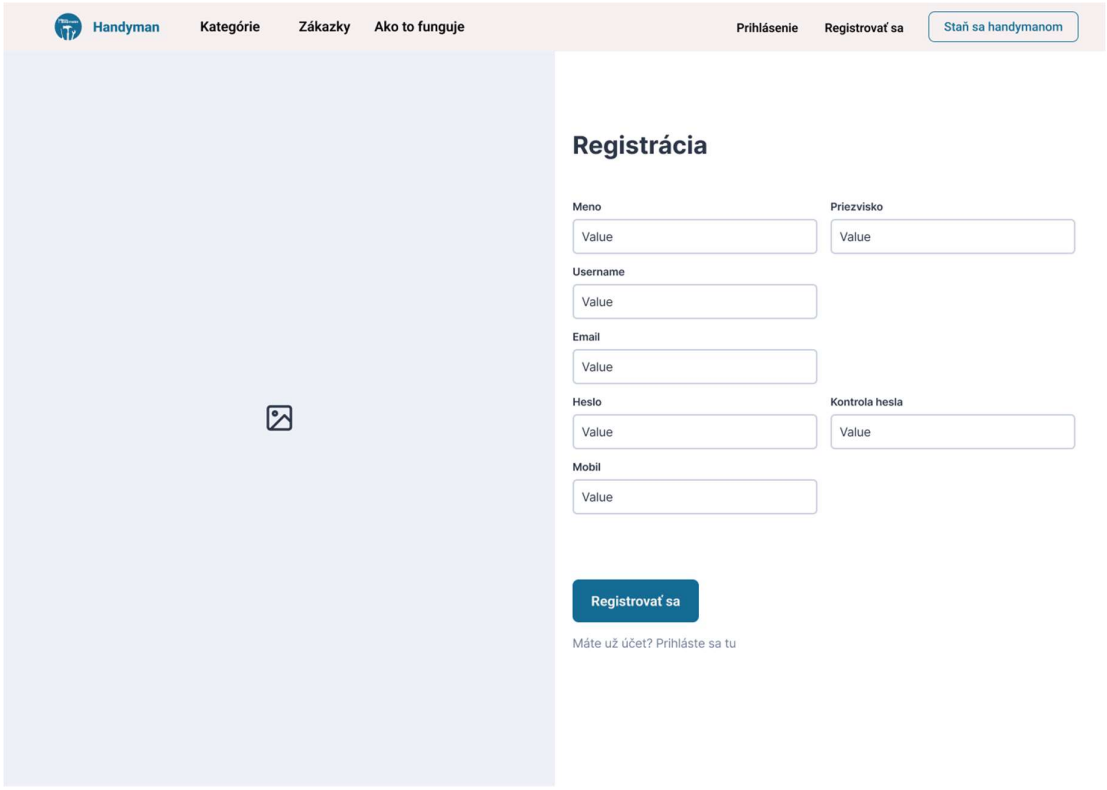
Obrazovka na prihlásenie bude navrhnutá s cieľom poskytnúť užívateľovi jednoduchý a intuitívny proces prihlásenia. Bude obsahovať polia na zadanie emailu a hesla, tlačidlo pre prihlásenie a odkaz na obnovu zabudnutého hesla. V prípade, že užívateľ zabudne svoje heslo, bude mať možnosť ho jednoducho zresetovať pomocou odkazu na obnovu hesla. Tento návrh má za cieľ minimalizovať frustráciu užívateľa pri procese prihlásenia a zabezpečiť, že môže ľahko získať prístup k svojmu účtu aj v prípade, že si nezapamätá svoje heslo.



Obrázok 8. Obrazovka na prihlásenie

3.2.3 Obrazovka na registráciu

Obrazovka na registráciu bude navrhnutá s cieľom poskytnúť novým užívateľom jasný a stručný proces vytvorenia účtu. Bude obsahovať polia na zadanie potrebných údajov, ako sú meno, priezvisko, heslo, potvrdenie hesla atď. Taktiež bude obsahovať tlačidlo na registráciu. Návrh tejto obrazovky je zameraný na zabezpečenie toho, aby proces registrácie bol pre užívateľa jednoduchý a intuitívny.

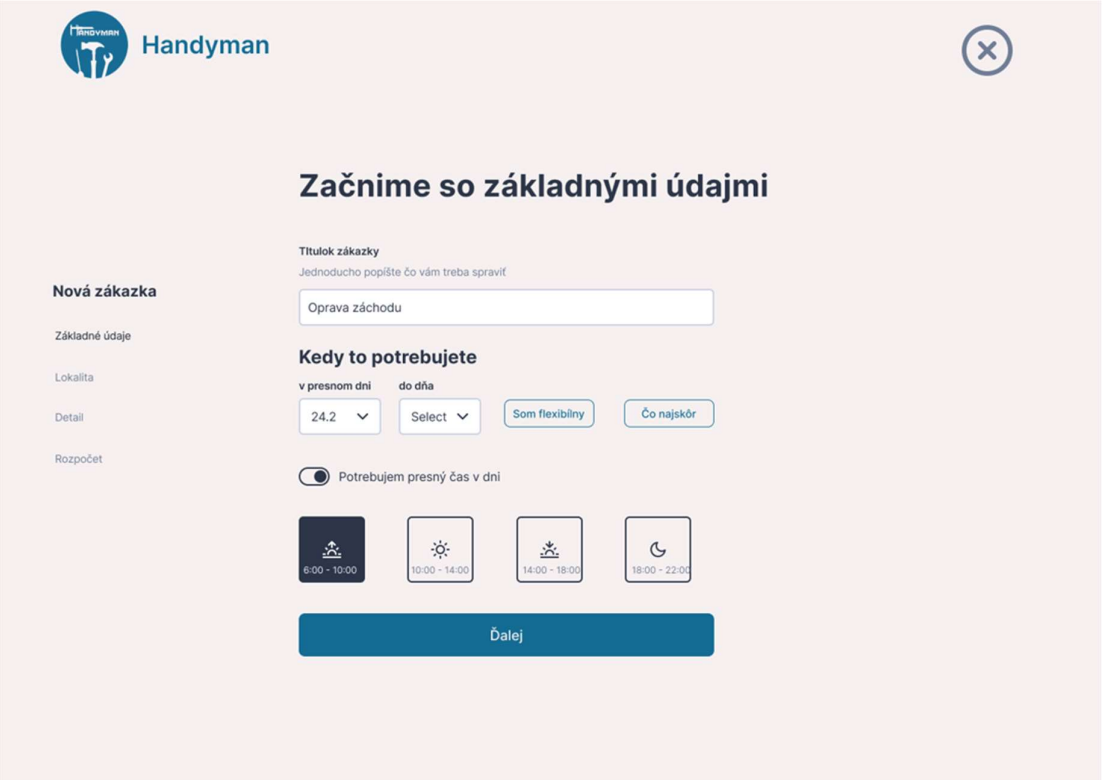


Obrázok 9. Obrazovka na registráciu

3.2.4 Stránka na pridanie novej zákazky

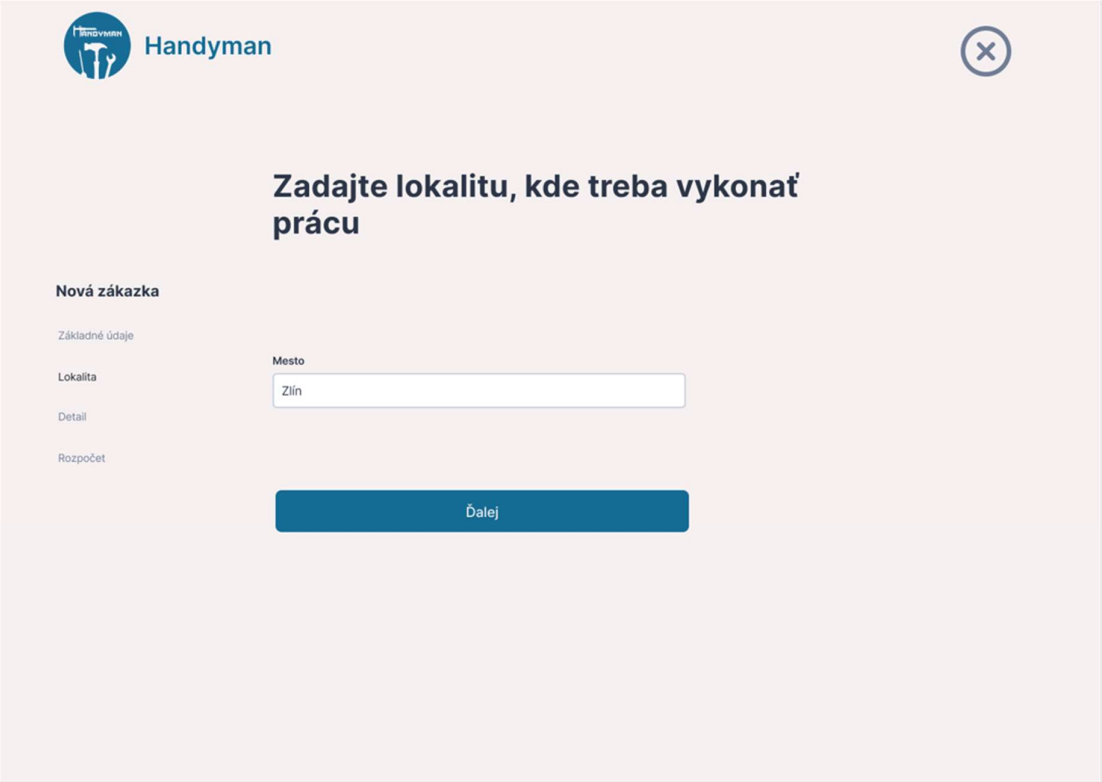
Stránka na pridanie novej zákazky je navrhnutá tak, aby umožnila užívateľom jednoduché a efektívne vytvorenie novej zákazky. Táto stránka sa skladá z niekoľkých obrazoviek, ktoré užívateľa postupne vedú procesom vytvárania zákazky.

Na prvej obrazovke je užívateľ vyzvaný, aby vyplnil základné informácie o zákazke, ako je jej názov, požadovaný časový rámec pre jej dokončenie alebo dátum, kedy by mal byť úkon vykonaný. Ak si užívateľ želá, môže tiež určiť konkrétne časové rozmedzie, kedy by mala byť úloha vykonaná.



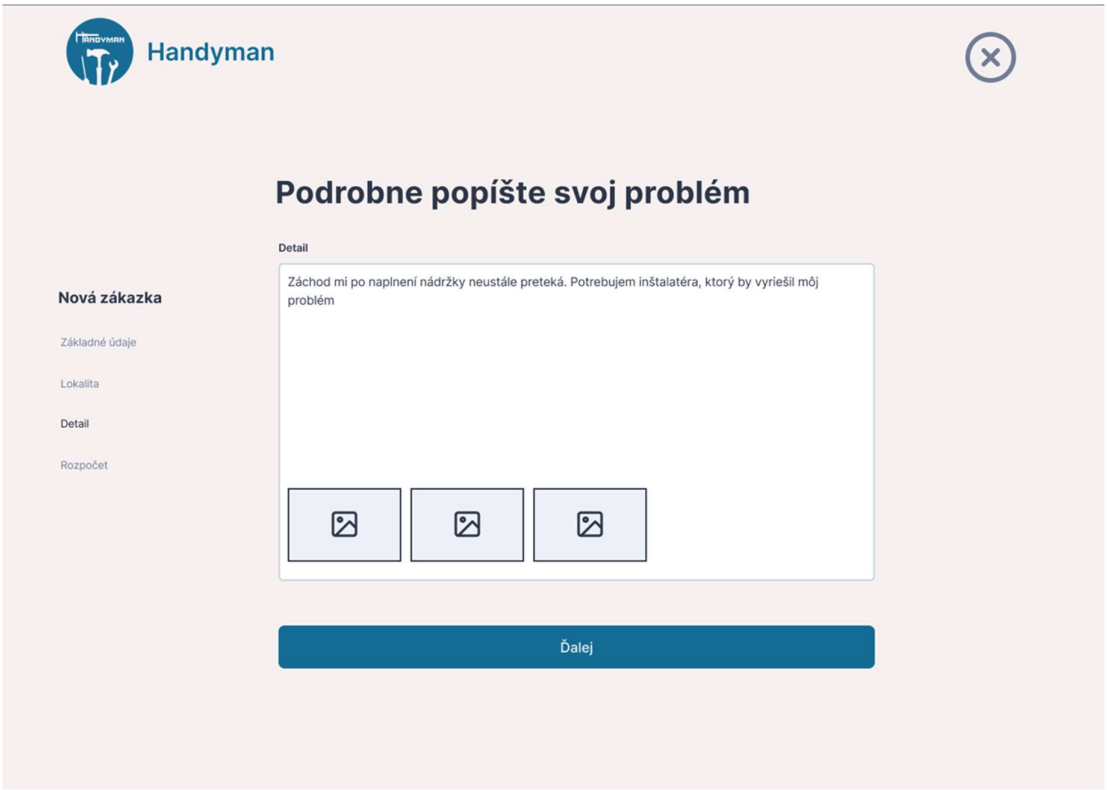
Obrázok 10. Nová zákazka – základné údaje

Na nasledujúcej obrazovke je užívateľ požiadaný o zadaní miesta, kde má byť zákazka vykonaná. Toto zahŕňa zadanie konkrétnej adresy alebo oblasti.



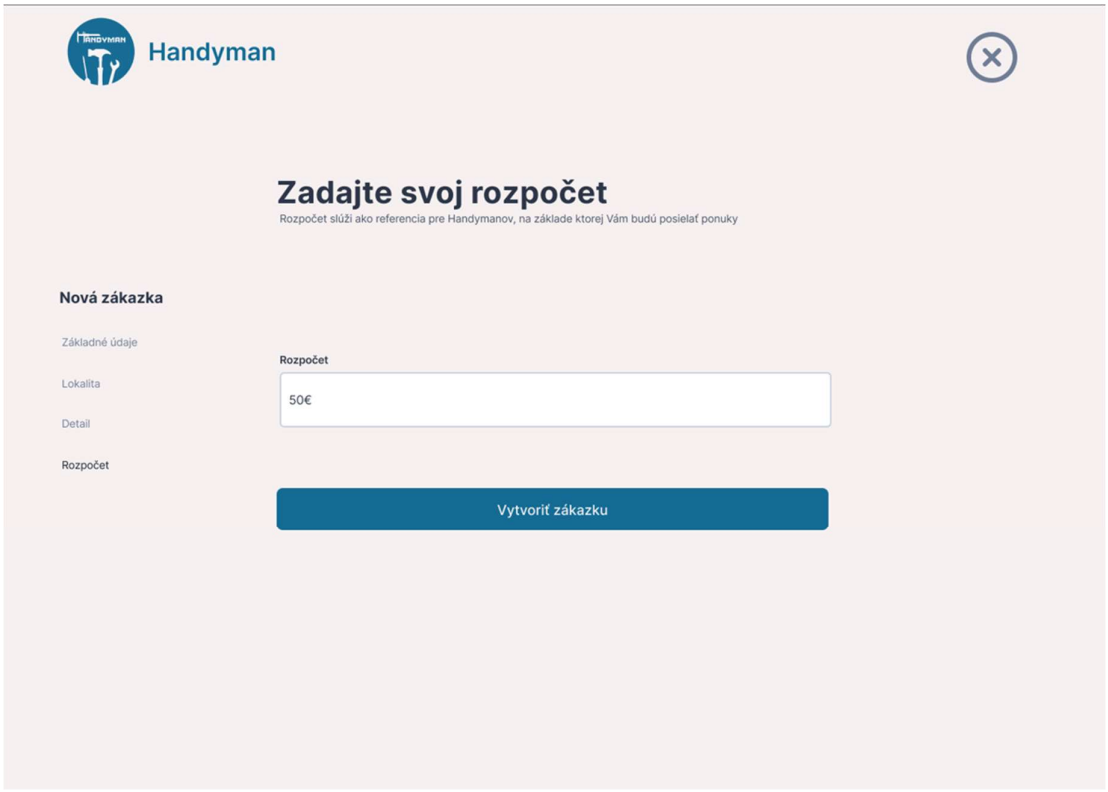
Obrázok 11. Nová zákazka – lokalita

Na tretej obrazovke má užívateľ možnosť podrobne opísať svoju zákazku, pričom môže pridať aj obrázky, ktoré by mohli pomôcť potenciálnym výkonným užívateľom lepšie pochopiť úlohu.



Obrázok 12. Nová zákazka - detail

Na poslednej obrazovke užívateľ uvedie svoj predpokladaný rozpočet pre zákazku a potom stlačí tlačidlo "Vytvoriť zákazku", ktoré uloží všetky informácie a pridá zákazku do zoznamu dostupných zákaziek. Tento návrh je zameraný na to, aby bol proces vytvárania zákazky jasný a intuitívny pre užívateľa.

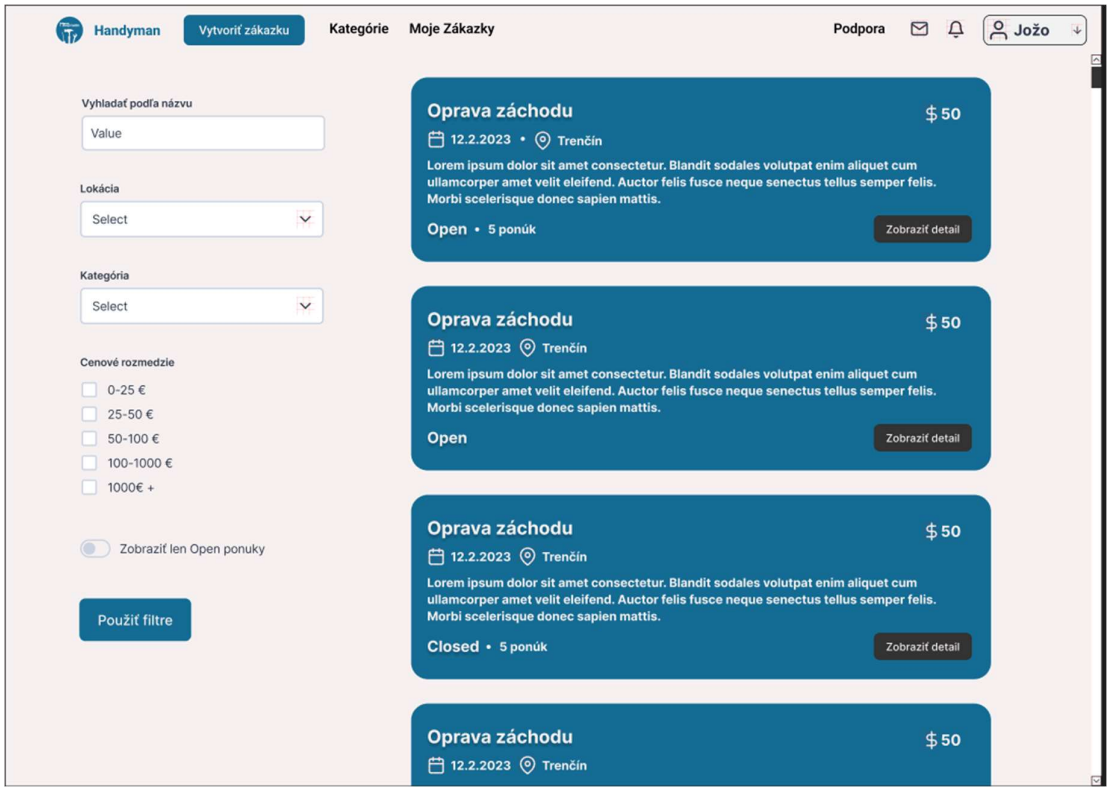


Obrázok 13. Nová zákazka - rozpočet

3.2.5 Obrazovka na zobrazenie zoznamu zákaziek

Obrazovka pre zobrazenie zoznamu zákaziek je navrhnutá tak, aby užívateľom poskytla jasný a prehľadný pohľad na dostupné zákazky. Každá zákazka v zozname zobrazuje názov, skrátený popis, lokalitu, rozpočet, dátum dokončenia, stav zákazky a počet ponúk. K dispozícii je aj tlačidlo pre zobrazenie detailov o zákazke.

Na ľavej strane obrazovky je panel s možnosťami filtrovania zákaziek. Užívatelia môžu filtrovať zákazky podľa názvu, lokality, kategórie, cenového ohodnotenia a môžu tiež vybrať možnosť zobraziť len otvorené zákazky. Tlačidlo "Použiť filtre" potom uplatní zvolené kritériá na zoznam zákaziek. Táto obrazovka je navrhnutá tak, aby umožnila užívateľom efektívne vyhľadávať a vyberať zákazky, ktoré najlepšie vyhovujú ich potrebám a preferenciám.

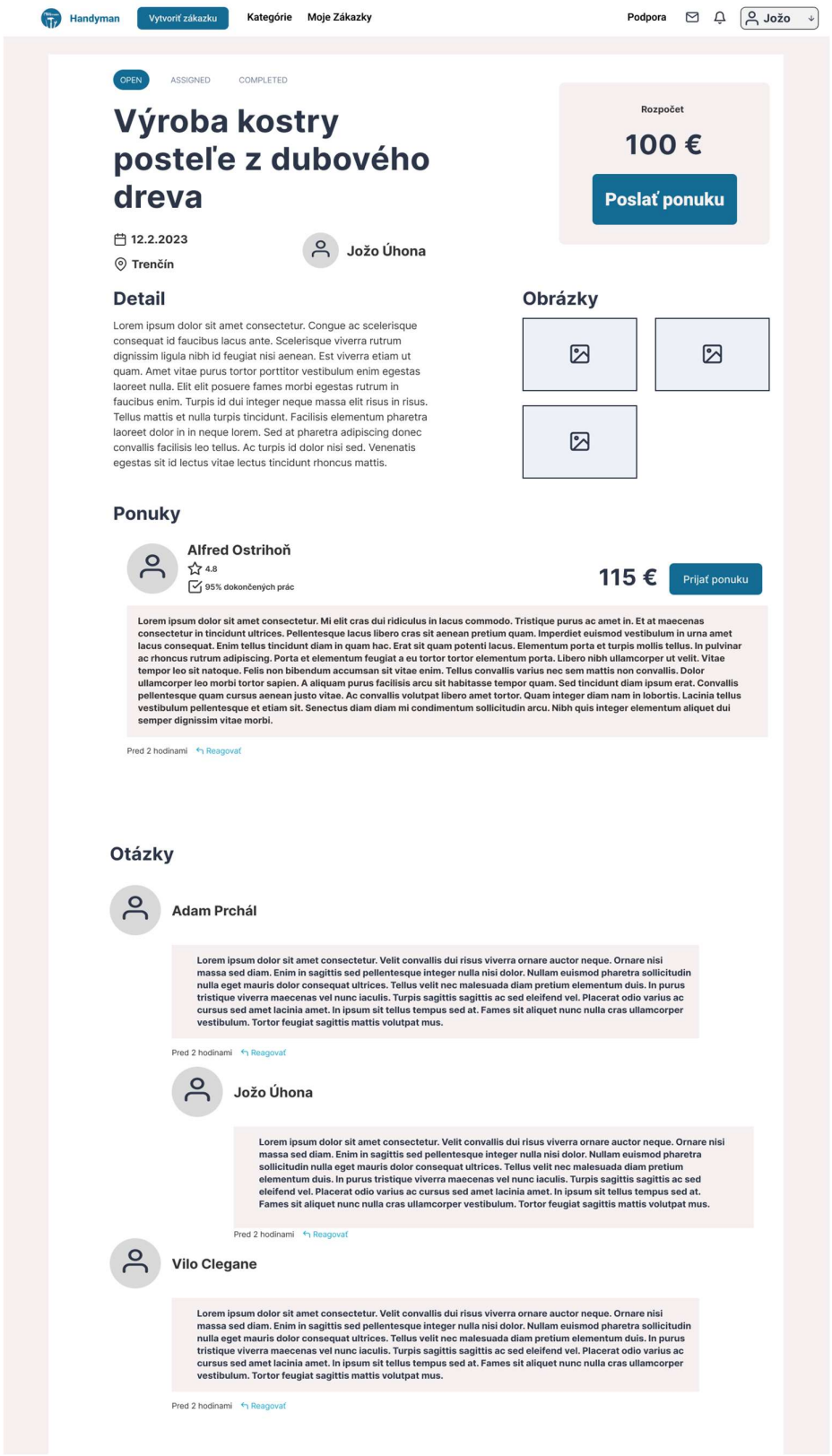


Obrázok 14. Zoznam zákaziek

3.2.6 Obrazovka na zobrazenie detailu zákazky

Detail zákazky je navrhnutý tak, aby poskytoval komplexný prehľad o konkrétnej zákazke. Na tejto obrazovke sú zobrazené všetky relevantné informácie o zákazke, vrátane ponúk od potenciálnych vykonávateľov a otázok, ktoré položili ostatní užívatelia ohľadom zákazky.

Okrem poskytovania informácií, táto obrazovka umožňuje interakciu medzi užívateľmi. Na ponuky a otázky môžu ostatní užívatelia reagovať formou komentárov, čím sa podporuje diskusia a výmena názorov. Majiteľ zákazky má na tejto obrazovke možnosť prijať ponuku, ktorá mu najviac vyhovuje, čím sa zabezpečuje efektívne a plynulé riešenie zákaziek. Navyše, užívatelia majú možnosť poslať svoje vlastné ponuky, čím sa podporuje konkurencia a umožňuje vykonávateľom predstaviť svoje služby v najlepšom svetle.



Obrázok 15. Detail Zákazky

4 VÝVOJ WEBOVEJ APLIKÁCIE

Táto časť bakalárskej práce sa zaoberá vývojom webovej aplikácie Handyman. Proces vývoja je rozčlenený na dve kľúčové časti - implementáciu back-endu a front-endu.

Back-end aplikácie, ktorý zahŕňa serverovú časť a databázu, je vytvorený pomocou technológií ako Node.js, Express.js a MongoDB. Tieto nástroje sú spolu použité na implementovanie API rozhrania, autentifikáciu, autorizáciu a iné.

Front-end aplikácia, ktorá zahŕňa užívateľské rozhranie a interakciu, je implementovaná pomocou JavaScriptovej knižnice React.js. Táto knižnica umožňuje vytvárať dynamické a responzívne užívateľské rozhrania.

Výsledkom tejto práce je prototyp aplikácie Handyman, ktorý predstavuje základnú funkcionalitu aplikácie. Tento prototyp slúži ako základ pre ďalší vývoj a testovanie funkcií, a umožňuje rýchle a efektívne iterácie vývoja.

4.1 Implementácia back-end časti

Táto sekcia sa zaoberá implementáciou back-end časti webovej aplikácie Handyman. Práca na back-ende zahŕňa niekoľko kľúčových aspektov: vytvorenie modelov, middleware, rozhraní, API trás (routes) a Controllerov.

Modely sú základným stavebným blokom aplikácie. Sú to štruktúry, ktoré definujú, ako sú údaje uložené v databáze. Sú vytvorené s využitím schém v Mongoose, ktoré poskytujú silný základ pre validáciu a správu v MongoDB. [22][23]

Middleware je súčasťou aplikačného stacku Express.js. Predstavuje sadu funkcií, ktoré sa vykonávajú predtým, než sa dosiahne konečný request handler. Middleware je použitý na spracovanie požiadaviek, manipuláciu s dátami a správu chýb. [20][21]

Rozhrania poskytujú silný nástroj na definovanie štruktúry dát a funkcionalít pre aplikáciu.

API trasy (routes) sú kľúčové pre poskytovanie prístupu k funkcionalite aplikácie. Sú definované pomocou Express.js a prestavujú konečné body (endpoints), na ktoré klient posiela požiadavky. Každá trasa je asociovaná s konkrétnym typom HTTP požiadavky (GET, POST, PUT, DELETE) a má priradené funkciu, ktorá sa má vykonať pri prijatí takejto požiadavky. [20][21]

Kontrolery sú súčasťou architektúry aplikácie, ktorá manipuluje s dátami na základe požiadaviek prijatých cez API trasy. Tieto funkcie zodpovedajú za spracovanie dát prijatých v požiadavke, komunikáciu s databázou pre vytváranie, čítanie, aktualizáciu a mazanie záznamov (CRUD operácie), a následne vracajú odpovede klientom. [33]

Dôkladná realizácia týchto komponentov je kľúčová pre správne fungovanie webovej aplikácie. Nasledujúce časti predstavia to, ako boli tieto komponenty vytvorené a implementované.

4.1.1 Vytvorenie Modelov

V tejto sekcii sú opísané modely, ktoré sú nevyhnutné pre správne fungovanie backendu webovej aplikácie. Pozornosť je venovaná nasledujúcim modelom: zákazka (contract), oblasť práce (fieldOfWork), užívateľ (user), profil užívateľa (userProfile), ponuka (offer), a otázka (question).

Model užívateľa

Model užívateľa definuje štruktúru informácií, ktoré sú o užívateľovi ukladané a spracovávané. Medzi tieto informácie patria osobné údaje ako meno, priezvisko, email, užívateľské meno, heslo, dátum narodenia, mobilné číslo a mesto. Taktiež model užívateľa uchováva informácie o dátume vytvorenia a poslednej aktualizácie účtu, o aktivite profilu a odkazy na zákazky a profil užívateľa. Tieto údaje umožňujú efektívnu správu užívateľských účtov a ich interakcií v rámci aplikácie.

Model zákazky

Model zákazky predstavuje štruktúru, ktorá obsahuje informácie týkajúce sa jednotlivých zákaziek vytvorených v aplikácii. Kľúčové atribúty modelu zákazky zahŕňajú názov, popis, dátum dokončenia, ponuky, otázky, stav, plat, oblasť práce, dátum vytvorenia, dátum poslednej úpravy, lokalitu a informácie o užívateľovi, ktorý zákazku vytvoril.

Názov a popis poskytujú podrobnosti o zákazke, zatiaľ čo atribút "doneIn" označuje dátum, kedy má byť zákazka dokončená. Atribút "offers" obsahuje referencie na ponuky priradené k danej zákazke, zatiaľ čo "questions" obsahuje otázky týkajúce sa zákazky. Stav zákazky je prednastavený na "Open" a označuje, či je zákazka otvorená pre ponuky alebo nie. "Salary" označuje očakávanú odmenu za dokončenie zákazky. Atribút "fieldOfWork" odkazuje na oblasť práce, do ktorej zákazka spadá, a "createdBy" uvádza užívateľa, ktorý zákazku vytvoril. "CreatedAt" a "updatedAt" sú dátumy vytvorenia a poslednej úpravy

zákazky a "location" uvádza geografickú lokalitu zákazky. Tieto atribúty dohromady umožňujú efektívne riadenie a sledovanie zákaziek v rámci aplikácie.

Model oblasť práce

Model oblasti práce predstavuje štruktúru, ktorá je zodpovedná za kategorizáciu rôznych zákaziek podľa typu práce. Základné atribúty tohto modelu sú identifikátor a názov.

Identifikátor (_id) je automaticky generovaný a jedinečný pre každú oblasť práce. Názov (name) predstavuje názov oblasti práce a je povinným a jedinečným atribútom. Unikátnosť tohto atribútu zabezpečuje, že každá oblasť práce je jedinečne identifikovateľná v rámci systému.

Tento model umožňuje užívateľom lepšie kategorizovať a hľadať zákazky podľa špecifických oblastí práce, čo zvyšuje efektivitu a používateľské pohodlie aplikácie.

Model ponuky

Model ponuky sa vytvára na základe informácií o jednotlivých ponukách, ktoré užívatelia predložia na zákazky v systéme. Každá ponuka obsahuje atribúty ako identifikátor, identifikátor zákazky, identifikátor užívateľa, cena a komentár.

Identifikátor (_id) je jedinečný pre každú ponuku a je generovaný automaticky. Identifikátor zákazky (contractId) a identifikátor užívateľa (userId) sú odkazy na zodpovedajúce modely zákazky a užívateľa a sú povinné.

Cena (price) a komentár (comment) sú atribúty, ktoré užívateľ uvádza pri vytváraní ponuky. Cena je suma, za ktorú je užívateľ ochotný vykonať zákazku, a komentár môže byť ďalšie vysvetlenie alebo podrobnosti týkajúce sa ponuky.

Tento model je kľúčový pre správnu interakciu medzi užívateľmi a zákazkami, keďže umožňuje užívateľom predkladať svoje ponuky na vykonanie zákaziek.

Model otázky

Model otázky je navrhnutý na uchovávanie otázok, ktoré užívatelia môžu klásť v súvislosti s konkrétnymi zákazkami. Každá otázka má nasledujúce atribúty: identifikátor, otázku, zákazku, dátum vytvorenia, poslednú aktualizáciu a autora.

Identifikátor (_id) je jedinečný pre každú otázku a je generovaný automaticky. Otázka (question) je text, ktorý užívateľ napíše pri kladení otázky.

zákazka (contract) je odkaz na zodpovedajúci model zákazky, ku ktorej sa otázka vzťahuje, a je povinný atribút.

Atribút dátumu vytvorenia (createdAt) a poslednej aktualizácie (lastUpdate) sú automaticky nastavované na aktuálny dátum a čas pri vytváraní alebo aktualizácii otázky.

Autor (createdBy) je odkaz na model užívateľa, ktorý otázku vytvoril. Tento atribút je povinný a slúži na identifikáciu užívateľa, ktorý otázku položil.

4.1.2 Vytvorenie kontrolerov

V rámci tejto časti práce sa bude diskutovať o vytvorení kontrolerov. Kontrolery sú esenciálnou súčasťou webovej aplikácie, keďže ovládajú tok dát medzi užívateľským rozhraním a databázou. Pre účely tejto práce boli vytvorené nasledujúce kontrolery: [33]

- Kontroler zákazky
- Kontroler užívateľa
- Kontroler autentifikácie
- Kontrolér pracovného zamerania

Každý z týchto kontrolerov má definované rôzne metódy pre manipuláciu s dátami príslušného modelu. Metódy zahŕňajú operácie ako vytváranie, čítanie, aktualizáciu a odstránenie dát (CRUD operácie), ale môžu obsahovať aj špecifické metódy pre konkrétne potreby aplikácie. Konkrétne implementácie a účely týchto metód budú podrobne popísané v nasledujúcich sekciách.

Kontroler užívateľa

Kontroler užívateľa, ako je zobrazený v súbore `userController.ts`, je súčasťou aplikácie, ktorá slúži na spracovanie požiadaviek spojených s údajmi užívateľa. V tomto konkrétnom prípade je kontroler definovaný pomocou piatich funkcií: `register`, `getUsers`, `deleteUser`, `updateUser` a `getUser`.

- Funkcia `register`: Táto funkcia vytvára nového užívateľa v systéme na základe údajov získaných z požiadavky klienta. Ak je registrácia úspešná, vracia odpoveď so stavovým kódom 201, ktorý označuje úspešné vytvorenie zdroja. V prípade chyby je vrátený stavový kód 400.

- Funkcia `getUsers`: Táto funkcia získava všetkých užívateľov z databázy a vracia ich ako odpoveď na klientovu požiadavku. V prípade, že sa nenájdu žiadny užívatelia bude vrátený stavový kód 204, V prípade chyby je vrátený stavový kód 500.
- Funkcia `deleteUser`: Táto funkcia odstraňuje užívateľa zo systému. Ak je užívateľ úspešne odstránený, vracia odpoveď s informáciou o úspešnom odstránení. V prípade chyby je vrátený stavový kód 500.
- Funkcia `getUser`: Táto funkcia získava a vracia údaje o konkrétnom užívateľovi na základe identifikátora užívateľa získaného z požiadavky klienta.

Tieto funkcie predstavujú základné operácie CRUD, ktoré sú kľúčové pre manipuláciu s databázovými zdrojmi.

Kontroler autentifikácie

Kontroler autentifikácie, ako je zobrazený v súbore `authController.ts`, obsahuje štyri funkcie: `loginUser`, `logoutUser`, `verifyToken` a `generateAccessToken`, ktoré sú nevyhnutné pre riadenie prístupu a identifikáciu užívateľov.

- Funkcia `loginUser`: Táto funkcia je zodpovedná za prihlásenie užívateľa do systému. Najprv sa skontroluje, či sa v tele požiadavky nachádza email a heslo. Ak nie bude odoslaný http status 400 so správou, že neboli zadané potrebné prihlasovacie údaje. Ak áno vyhľadá sa užívateľ pomocou emailu, ak sa užívateľ nenašiel odošle sa http status 403, ktorý oboznámi užívateľa, že zadal nesprávne heslo alebo email. Ak sa našiel užívateľ, porovná sa zadané heslo užívateľom a heslom, ktoré patrí účtu daného emailu. Ak sa heslá nezhodujú odošle sa znova http status 403. V prípade akejkoľvek chyby na serveri bude odoslaný http status 500. následne sa vytvorí prístupový token a obnovovací token pomocou balíčka JWT. Tieto tokeny sa uložia do cookies, ktoré sú odoslané späť ku klientovi. Ak všetko prebehlo v poriadku odošle sa http status 200.

```
export const loginUser = async (req: MyRequest, res: MyResponse) => {

  if (!req.body || !req.body.email || !req.body.password) {
    return res.status(400).json({ message: 'Missing email or password' })
  }
  let user
  try {
   user = await UserModel.findOne({ email: req.body.email }).exec()
    if (user === null) {
      return res.status(403).json({ message: "wrong email or password" })
    }
    const correctPassword = await bcrypt.compare(req.body.password, user.password)
    if (!correctPassword) {
      return res.status(403).json({ message: 'wrong email or password' })
    }
  } catch (err:any) {
    return res.status(500).json({message: err.message})
  }

  const verifiedUser: object = {
    '_id': user._id,
    'slug': user.slug,
    'firstName': user.firstName,
    'lastName': user.lastName,
    'email': user.email,
    'userName': user.userName,
    'contracts': user.contracts,
    'profile': user.profile,
  }

  const refreshTokenSecret = process.env.REFRESH_TOKEN_SECRET

  if (!refreshTokenSecret) {
    throw new Error('REFRESH_TOKEN_SECRET is not defined')
  }

  const secret: Secret = refreshTokenSecret
  const accessToken = generateAccessToken(verifiedUser)
  const refreshToken = jwt.sign(verifiedUser, secret)

  res.cookie('accessToken', accessToken, {
    httpOnly: true,
    secure: true,
    sameSite: 'none', // make strict before deploying app for production
    maxAge: 60 * 60 * 1000, //60 minutes
  })

  res.cookie('refreshToken', refreshToken, {
    httpOnly: true,
    secure: true,
    sameSite: 'none', // make strict before deploying app for production
    maxAge: 7 * 24 * 60 * 60 * 1000, // 7 days
  })

  res.status(200).json({ message: 'Login successful' })
}
```

Obrázok 16. Ukážka kódu: loginUser

- Funkcia `logoutUser`: Táto funkcia odstraňuje prístupový token a obnovovací token z cookies, čím sa užívateľ odhlási zo systému.
- Funkcia `verifyToken`: Táto funkcia overuje platnosť prístupového tokenu, ktorý je získaný z cookies. Ak je token platný, funkcia vracia stav autentifikácie a údaje dekódované z tokenu. Ak token nie je platný alebo chýba, funkcia vracia stav autentifikácie ako false.

```
export const verifyToken = async (req:MyRequest, res:MyResponse) => {
  const accessToken = req.cookies.accessToken

  if (!accessToken) {
    return res.status(401).json({ isAuthenticated: false })
  }
  const accessTokenSecret=process.env.ACCESS_TOKEN_SECRET
  if(!accessTokenSecret){
    throw new Error('ACCESS_TOKEN_SECRET is not defined')
  }
  const secret:Secret=accessTokenSecret
  try {
    const decoded = jwt.verify(accessToken, secret)
    res.status(200).json({ isAuthenticated: true, user: decoded })
  } catch (error) {
    console.error('Error verifying token:', error)
    res.status(401).json({ isAuthenticated: false })
  }
}
```

Obrázok 17. Ukážka kódu: verifyToken

- Funkcia `generateAccessToken`: Táto pomocná funkcia je použitá na generovanie prístupového tokenu. Vstupom je objekt užívateľa a výstupom je prístupový token, ktorý platí po dobu 15 minút.

Tieto funkcie poskytujú základné nástroje pre správu autentifikácie užívateľov a zabezpečenie prístupu k dátam v systéme.

Kontroler zákazky

Kód v súbore contractController.ts definuje rôzne funkcie, ktoré spravujú zákazky v systéme. Tieto funkcie zahŕňajú:

- getContracts: Táto funkcia zodpovedá za načítanie všetkých zákazok z databázy a ich následné odoslanie ako odpoveď klientovi.
- getContract: Táto funkcia umožňuje získať konkrétnu zákazku identifikovanú pomocou jedinečného identifikátora.
- updateContract: Táto funkcia umožňuje aktualizovať existujúcu zákazku. Príchodzí údaje sú použité na úpravu záznamu existujúcej zákazky v databáze.
- createContract: Táto funkcia umožňuje vytvoriť novú zákazku. Tvorba novej zákazky je spojená s konkrétnym používateľom, ktorý je identifikovaný pomocou ID užívateľa, a nová zákazka je potom pridaná do databázy a do zoznamu zákazok používateľa.
- deleteContract: Táto funkcia umožňuje vymazať existujúcu zákazku. Vymazanie zákazky je možné len v prípade, ak používateľ, ktorý požiadal o vymazanie, je rovnaký používateľ, ktorý zákazku vytvoril.

Tieto funkcie sú potom priradené k príslušným trasám API a spracovávajú prichádzajúce požiadavky od klientov. Tieto funkcie poskytujú plnú správu nad životným cyklom zákaziek v systéme, od ich vytvorenia po aktualizáciu a vymazanie.

Kontroler pracovného zamerania

Súbor `fieldOfWorkController.ts` obsahuje súbor funkcií, ktoré umožňujú manipuláciu s údajmi v kolekcii `FieldOfWorkModel`, ktorá reprezentuje rôzne pracovné zamerania v systéme. Tieto funkcie sú nasledovné:

- fieldOfWorksDelete: Táto funkcia je zodpovedná za odstránenie existujúceho záznamu o pracovnom zameraní. Ak v procese vznikne chyba, vráti sa HTTP status 500 spolu s chybovou správou.
- fieldOfWorksUpdate: Táto funkcia aktualizuje existujúci záznam o pracovnom zameraní. Ak je v tele požiadavky zadané nové meno, zmení sa aktuálna hodnota mena v modeli `FieldOfWorkModel`. Ak v procese vznikne chyba, vráti sa HTTP status 500 spolu s chybovou správou.

- fieldOfWorksCreate: Táto funkcia vytvára nový záznam o pracovnom zameraní v systéme. Ak v procese vznikne chyba, vráti sa HTTP status 500 spolu s chybovou správou.
- fieldOfWorksGetOne: Táto funkcia zasiela klientovi jeden záznam o pracovnom zameraní.
- fieldOfWorksGetAll: Táto funkcia načíta a zasiela všetky záznamy o pracovných zameraniach v systéme. Ak v procese vznikne chyba, vráti sa HTTP status 500 spolu s chybovou správou.

Všetky tieto funkcie sú priradené k príslušným trasám API a sú navrhnuté tak, aby spracovávali prichádzajúce požiadavky od klientov. Poskytujú plnú kontrolu nad životným cyklom údajov o pracovných zameraniach v systéme.

4.1.3 Vytvorenie Middleware funkcií

V rámci tejto časti práce sa bude diskutovať o vytvorení middleware funkcií. Middleware funkcie sú veľmi dôležitou súčasťou architektúry aplikácie. Sú to funkcie, ktoré sa spustia medzi príchodom požiadavky a odoslaním odpovede. Middleware dokáže modifikovať requesty a response, vykonať akékoľvek potrebné operácie alebo dokonca zastaviť ďalšie spracovanie a priamo odoslať odpoveď. Pre účely tejto práce boli vytvorené nasledujúce middleware funkcie:

- authenticationOfUser : Táto middleware funkcia je zodpovedná za overenie autentifikácie užívateľa. Začína tým, že získa prístupový token (accessToken) z cookies požiadavky. Potom kontroluje, či je definovaný secret pre tokeny, ktorý je uložený v premenných prostredia ako ACCESS_TOKEN_SECRET. Ak nie je prístupový token prítomný, funkcia vráti HTTP status 401 (Neautorizovaný) spolu so správou "Prístup zamietnutý. Užívateľ nie je autentifikovaný". Ak je prístupový token prítomný, funkcia sa pokúsi overiť jeho platnosť pomocou funkcie verify z balíčka jsonwebtoken. Ak je token platný, informácie o užívateľovi sa pridajú do objektu požiadavky (req.user = verified) a potom sa zavolá funkcia next(), aby sa pokračovalo v spracovaní ďalšej middleware funkcie. Ak je token neplatný, funkcia vráti HTTP status 400 (Nesprávna požiadavka) so správou "Neplatný token".
- getFieldOfWorkById: Táto middleware funkcia je zodpovedná za získanie konkrétneho zamerania práce pomocou identifikačného čísla. Ak je vyhľadanie

v databáze úspešné, vráti sa do odpovede dané zameranie práce a zavolá sa funkcia next(). Ak nie, tak sa vráti HTTP status 404, ktorý znamená že neexistuje záznam v databáze, ktorá by mala rovnaké identifikačné číslo.

- getPassword: Táto middleware funkcia je zodpovedná za spracovanie a zabezpečenia hesla užívateľa. Funkcia začne tým, že skontroluje dĺžku hesla. Ak je heslo kratšie ako 8 znakov, funkcia vráti chybovú správu „short password“. Ak je heslo dostatočne dlhé, vytvorí sa „salt“, čo je náhodná sekvencia, ktorá sa pridáva k heslu pre zvýšenie bezpečnosti. Tento „salt“ je následne použitá spolu s pôvodným heslom na vytvorenie hashovanej verzie hesla. Toto hashované heslo je následne uložené naspäť do prichádzajúcej požiadavky. Ak nastane chyba počas tohto procesu, funkcia vráti HTTP status 500 a chybovú hlášku.
- GetUserById : Táto middleware funkcia funguje podobne ako funkcia getFieldOfWorkByID, kde sa pomocou identifikačného čísla vyhľadáva užívateľ v databáze.

4.1.4 Vytvorenie rozhrania

Rozhrania, alebo interfaces, sú dôležitým konceptom v mnohých programovacích jazykoch, vrátane TypeScriptu, ktorý je použitý v tejto aplikácií. Rozhrania definujú štruktúru objektov a umožňujú typovú kontrolu, čo znamená, že je možné skontrolovať, či sú objekty v správnom formáte ešte predtým, ako sa s nimi začne pracovať. [32]

Rozhrania „MyRequest“ a „MyResponse“ sú kľúčové pre zabezpečenie korektnosti typových požiadaviek a odpovedí v rámci back-end časti webovej aplikácie Handyman. Tieto rozhrania rozširujú štandardné „Request“ a „Response“ rozhrania, ktoré sú poskytnuté v rámci knižnice Express.js. Tieto rozšírenia umožňujú pridať dodatočné vlastnosti a metódy, ktoré môžu byť špecifické pre aplikáciu.

4.1.5 API Routes

API Routes, tiež známe ako trasy, sú kľúčové pre správne fungovanie webovej aplikácie. Predstavujú „trasy“ na serveri, ktoré sú naviazané na konkrétne funkcie. Tieto funkcie sa spustia, keď je daná trasa zasiahnutá požiadavkou od klienta. Každá trasa je asociovaná s konkrétnym typom HTTP požiadavky (GET,PUT,POST,DELETE) a má priradenú funkciu, ktorá sa má vykonať pri prijatí takejto požiadavky. [33]

4.2 Implementácia front-end časti

Implementácia front-end časti webovej aplikácie Handyman je realizovaná pomocou populárnej JavaScriptovej knižnice React.js a utility-first CSS frameworku Tailwind.css. Obe tieto technológie sú široko používané vo vývoji webových aplikácií pre ich flexibilitu, výkonnosť a efektívnu prácu so štýlmi. [7]

Nasledujúce podsekcie sa budú zaoberať detailnejším popisom jednotlivých častí implementácie:

- Routes – V React.js aplikáciách sú trasy vytvárané s použitím react-router knižnice. Trasy definujú, ktoré komponenty majú byť zobrazené v závislosti od aktuálnej URL cesty. [7]
- Komponenty - sú jednotlivé stavebné bloky react.js aplikácie. Každý komponent môže obsahovať svoj vlastný stav, metódy a môže byť štylizovaný pomoocu Tailwind.css. [7]
- Pages – sú špeciálne komponenty, ktoré predstavujú celé stránky vo webovej aplikácii. Každá stránka je asociovaná s konkrétnou trasou a obsahuje ostatné komponenty, ktoré tvoria obsah tejto stránky. [7]
- Rozhrania – sú použité na definovanie štruktúr dát a pomáhajú pri zachované typovej bezpečnosti a prehľadnosti kódu. [32]

4.2.1 Routes

Routes, alebo trasy, sú esenciálnou časťou každej single-page aplikácie ako je aj webová aplikácia Handyman. Definujú navigáciu medzi jednotlivými komponentami alebo stránkami aplikácie na základe URL cesty. Ako súčasť implementácie sú trasy v Handyman aplikácii rozdelené na dva druhy: public a private.

- Public trasy sú prístupné užívateľom, ktorí nie sú prihlásení. Medzi tieto trasy patrí domovská stránka, stránka pre registráciu alebo stránka pre prihlásenie.
- Private trasy sú prístupné iba pre prihlásených užívateľov. Patrí tu napríklad domovská stránka pre prihlásených užívateľov, stránka pre pridávanie nových kontraktov, stránka so zoznamom kontraktov a detail jednotlivých zákaziek.

Zabezpečenie prístupu k privátnym trasám je realizované pomocou funkcie „PrivateRoutes“. Táto funkcia skontroluje, či je užívateľ prihlásený pomocou overenia tokenu a na zá-

klade toho rozhodne, či mu umožní prístup k danej trase, alebo ho presmeruje na prihlasovaciu stránku. Taktiež odosiela údaje o prihlásenom užívateľovi svojím child routes.

```
import React, { useEffect, useState } from 'react'
import { Navigate, Outlet, useOutletContext } from 'react-router'
import { api } from '../API'

const  PrivateRoutes = () => {
  const [auth, setAuth] = useState(false);
  const [loading, setLoading] = useState(true);
  const [user,setUser]=useState([])
  useEffect(() => {
    const checkAuthentication = async () => {
      try {
        const response = await api.get('/auth/verify-token');
        if (response.status === 200) {
          setAuth(response.data.isAuthenticated);
          setUser(response.data.user)
        } else {
          setAuth(false);
        }
      } catch (error) {
        console.error('Error checking authentication:', error);
        setAuth(false);
      } finally {
        setLoading(false);
      }
    };
    checkAuthentication();
  }, []);

  if (loading) {
    return <div>Loading...</div>;
  }
  return (
    auth ? <Outlet context={[user, setUser]}/> : <Navigate to="/login" />
)
};

export default PrivateRoutes;
```

Obrázok 18. Ukážka kódu: PrivateRoutes

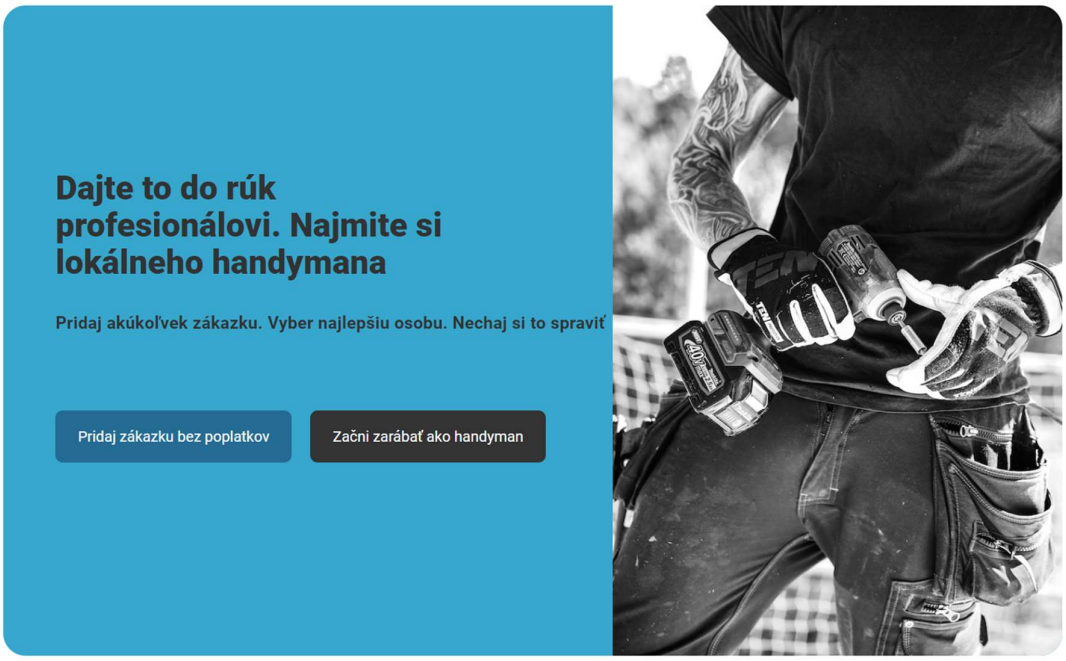
4.2.2 Pages

V tejto sekcií budú popísané nasledujúce stránky (Pages): domovská stránka pre neprihláseného užívateľa, Prihlasovacia stránka, stránka so zoznamom zákaziek, stránka s detailom zákazky, stránka na registráciu, stránka na pridanie zákazky. Vytvorené stránky budú referencie na návrhy vytvorené pomocou Figmy a však treba dodať, že niektoré stránky nie sú podobne návrhu a to z dôvodu časovej náročnosti.

Domovská stránka pre neprihláseného užívateľa

Táto stránka slúži ako vstupný bod webovej aplikácie, ktorá ma za úlohu ukázať užívateľovi aké služby ponúka a angažovať ho k tomu aby sa registroval. Skladá sa z nasledujúcich komponentov:

- Hero sekcia je prvotný komponent, ktorý má za úlohu zaujať užívateľa. Obsahuje Nadpis, podnadpis, obrázok a CTA tlačidlá, ktoré presmerujú užívateľa na registráciu alebo na stránku, ktorá oboznámi užívateľa ako začať pracovať ako handyman.
- HowItWorks sekcia je komponent, ktorá oboznámi užívateľa zjednodušeným priebehom od vytvorenia zákazky až po jej dokončenie.
- Testimonial sekcia je komponent, ktorý obsahuje výrok tretej strany o aplikácií, ktorá má za úlohu zvýšiť dôveru užívateľa a motivovať ho využiť služby webovej aplikácie Handyman

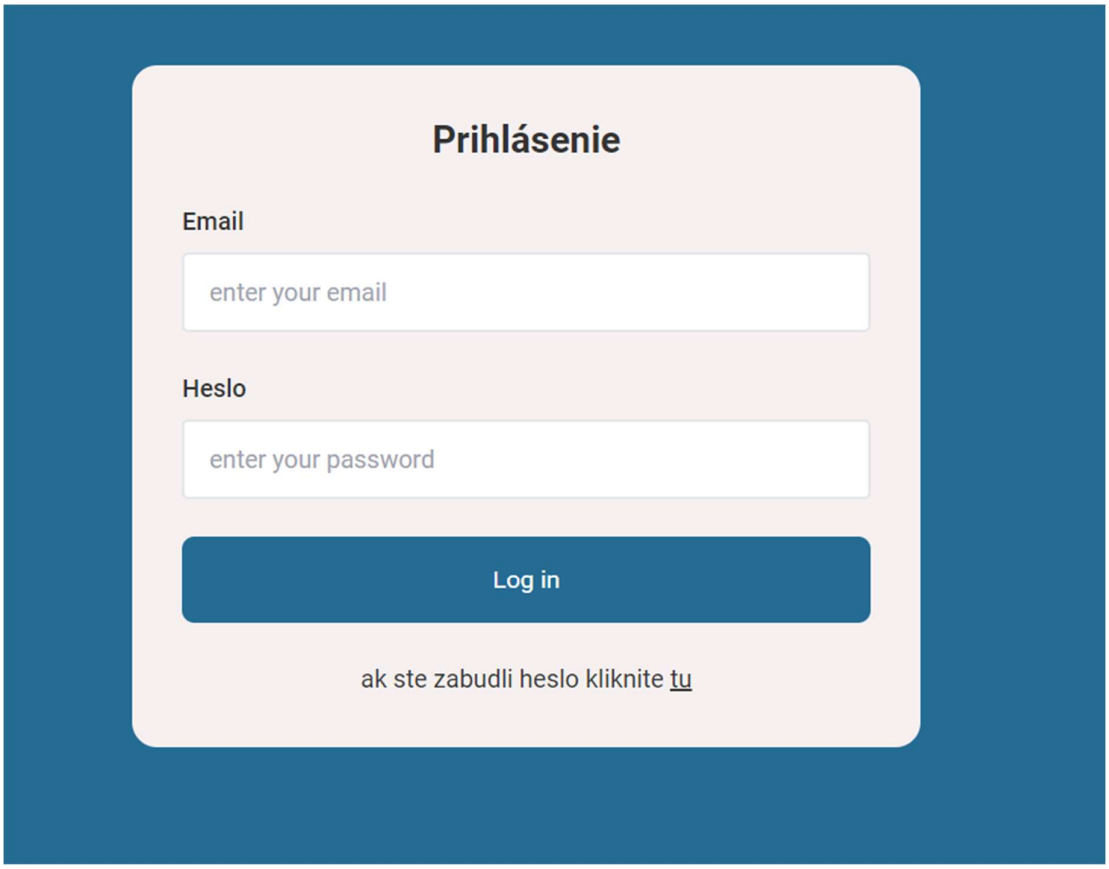


Obrázok 19. Komponent - Hero sekcia

Obrázok 20. Komponent - Popis služby

Stránka na prihlásenie

Stránka na prihlásenie umožňuje užívateľom prihlásiť sa do systému pomocou svojich prihlasovacích údajov email a heslo. Pri odoslaní formulára sa tieto údaje pošlú na server pomocou API na overenie. Ak je prihlásenie úspešné, užívateľa presmeruje na domovskú stránku ak nie, zobrazí sa chybová hláška.



Obrázok 21. formulár na prihlásenie

Stránka s detailom zákazky

Stránka pre detail zákazky (ContractDetailPage) je komponent, ktorý zobrazuje detailné informácie o konkrétnej zákazke. Tieto informácie zahŕňajú popis zákazky, termín dokončenia, ponuky, otázky a ďalšie podrobnosti týkajúce sa zákazky.

Komponent využíva identifikátor zákazky z parametrov URL a pomocou tohto identifikátora načíta podrobnosti o zákazke z API.

Po úspešnom načítaní dát sa zobrazia jednotlivé podkomponenty, ktoré zobrazujú rôzne časti detailov zákazky - status zákazky, hlavičku, podrobný popis, ponuky a otázky.

Obrázok 22. Aplikácia - Detail zákazky

Stránka zákaziek

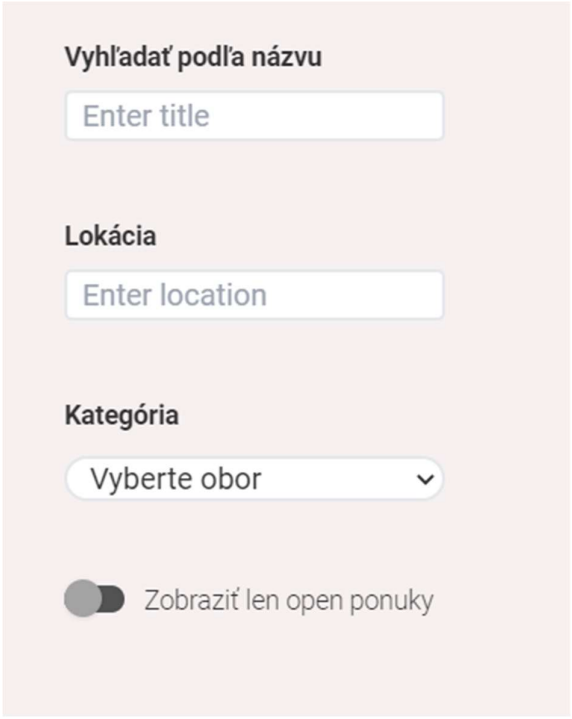
Stránka zákaziek (ContractsPage) je komponent, ktorý zobrazuje zoznam všetkých dostupných zákaziek. Tento komponent načíta zákazky a kategórie z API a potom ich zobrazuje pomocou komponentu ContractCard. Užívateľ má tiež možnosť filtrovať zákazky podľa názvu, lokality, vybranej kategórie a podľa toho, či sú zákazky stále otvorené.

Zoznam zákaziek je filtrovaný v reálnom čase podľa stavu premenných na filtrovanie (title, location, chosenCategory, onlyOpenOffers). Tieto stavy sú nastavené pomocou komponentu SearchFilter, ktorý poskytuje užívateľovi formulár pre nastavenie filtrov. Stavy sú "lifted up", teda sú udržiavané v rodičovskom komponente (ContractsPage), takže zmeny v týchto stavoch môžu ovplyvniť zobrazenie zákazok v komponente ContractsPage.

```
const filteredContracts : Contract[]  = contracts.filter(contract : Contract  => {
  return (
    contract.title.includes(title) &&
    contract.location.includes(location) &&
    (chosenCategory === '' || contract.fieldOfWork === chosenCategory) &&
    (!onlyOpenOffers || contract.status === 'Open'))
})
```

Obrázok 23. Funkcia na filtrovanie zákaziek

Po načítaní údajov z API sa zobrazia zákazky, ktoré spĺňajú aktuálne nastavené filtre. Ak sú nejaké zákazky dostupné, pre každú sa vytvorí nový komponent ContractCard, ktorý zobrazí podrobnosti o danej zákazke. Ak neexistujú žiadne zákazky, ktoré by spĺňali filtre, nebude sa zobrazovať žiadna ContractCard.

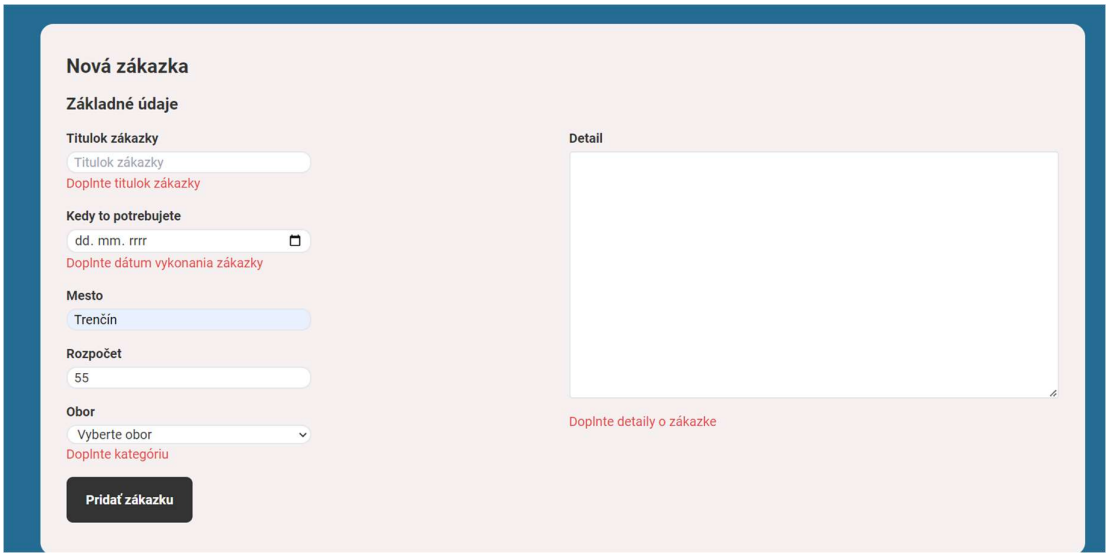


Obrázok 24. Komponent na filtrovanie zákaziek

Obrázok 25. Komponent - Kartička zákazky

Stránka na pridanie novej zákazky

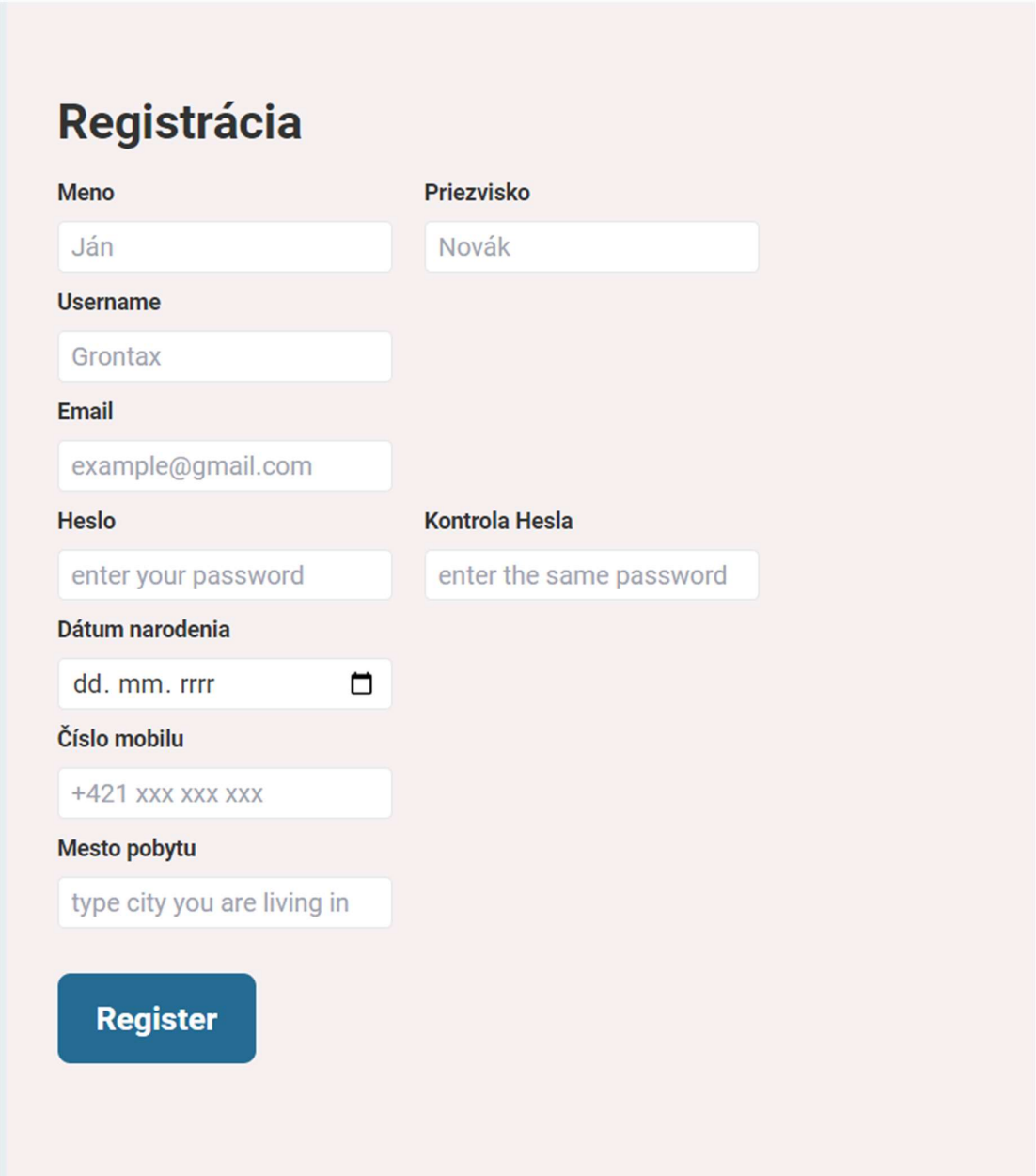
Táto stránka umožňuje užívateľom pridávať nové zákazky. Na tejto stránke je formulár, ktorý užívateľ vyplní informáciami o novej zákazke, ako je názov, termín, miesto, rozpočet a kategória. Taktiež stránka obsahuje validáciu vstupov, ktorá zabezpečuje, že všetky polia boli správne vyplnené. Po odoslaní formulára sa vykoná POST požiadavka na server prostredníctvom API s údajmi o novej zákazke. Je nutné dodať že UI je odlišné od UI navrhnuté pomocou Figmy a to z dôvodov časovej náročnosti implementovať navrhované UI a priority predstaviť funkcionalitu webovej aplikácie. V budúcich etapách vývoja webovej aplikácie bude UI prerobené tak, aby bolo v súlade s daným návrhom.



Obrázok 26. Formulár na pridanie zákazky

Stránka na registráciu

Táto stránka umožňuje registráciu užívateľov. Obsahuje formulár, do ktorej užívateľ vpíše svoje údaje ako meno, priezvisko, heslo, email atď. Taktiež obsahuje validáciu vstupov, ktorá zaručí to, aby sa užívateľ registroval korektne. Po odoslaní formulára sa skontroluje či je formulár validný, ak áno zavolá sa POST požiadavka pomocou API s údajmi o novom užívateľovi.



Obrázok 27. Formulár na registráciu

5 ZABEZPEČENIE WEBOVEJ APLIKÁCIE

Zabezpečenie webovej aplikácie Handyman predstavuje komplexný proces, ktorý zahŕňa viaceré vrstvy ochrany s cieľom predchádzať rôznym typom kybernetických útokov. Nasledujúce bezpečnostné opatrenia boli implementované do webovej aplikácie:

- Autorizácia a autentifikácia prostredníctvom JWT
- Implementácia express rate limiteru
- Využitie helmet balíčka
- Implementácia CORS politiky

5.1 Autorizácia a autentifikácia prostredníctvom JWT

JSON Web Token (JWT) je technológia založená na otvorenom štandarde , ktorý definuje kompaktný a sebaobsažný spôsob bezpečného prenosu informácií medzi dvoma stranami prostredníctvom JSON objektu. Tieto informácie môžu byť overené a dôveryhodné, pretože sú digitálne podpísané. JWT môže byť podpísaný pomocou tajného kľúča alebo pomocou pár kľúčov verejný/súkromný. [37]

V kontexte webovej aplikácie Handyman sa JWT používa pre autentifikáciu a autorizáciu. Po úspešnom prihlásení sa užívateľovi vytvorí a priradí jedinečný JWT. Tento token obsahuje dôležité informácie o užívateľovi a je časovo obmedzený, čo znamená, že po uplynutí určitej doby stráca svoju platnosť. Tento mechanizmus zabraňuje neoprávnenému prístupu k užívateľským údajom aj v prípade, že by bol token kompromitovaný.

Token sa potom používa na overenie identity užívateľa pri každej interakcii so serverom, ktorá vyžaduje autorizáciu. Napríklad, keď sa užívateľ pokúsi o prístup na stránky, ktoré sú dostupné len pre prihlásených užívateľov.

V Handyman aplikácii sa JWT token uchováva v tzv. http only cookies. Tieto cookies sú také, ktoré nie sú prístupné prostredníctvom JavaScriptu v prehliadači. Tým sa znižuje riziko tzv. cross-site scripting (XSS) útoku, kedy by útočník mohol skúsiť získať prístup k tokenu prostredníctvom skriptu vloženého do stránky. Toto riešenie zvyšuje bezpečnosť aplikácie a chráni užívateľské údaje. [38]

5.2 Implementácia express rate limiteru

S cieľom zabezpečiť webovú aplikáciu Handyman proti útokom typu brute-force a DDoS, bola implementovaná technológia Express Rate Limiter. Tento middleware, vstupný bod v komunikácii medzi klientom a serverom, funguje na princípe obmedzenia počtu požiadaviek, ktoré môžu byť odoslané z jednej IP adresy na server v určitej časovej jednotke. [39] [40]

Útoky typu brute-force alebo DDoS sú typické vysokým počtom požiadaviek smerujúcich na server za veľmi krátky časový interval, čo môže spôsobiť preťaženie a následnú nefunkčnosť servera. Express Rate Limiter týmto útokom čelí tým, že nastaví maximálny limit požiadaviek, ktoré môže daná IP adresa poslať za určený časový interval. Ak je tento limit prekročený, nasledujúce požiadavky sú automaticky zamietnuté až do uplynutia určenej časovej jednotky. [39] [40]

Takýto prístup prispieva k ochrane servera a zaisťuje jeho stabilné fungovanie aj pri pokusoch o útok. Okrem zvýšenej bezpečnosti servera tiež zabezpečuje, že legitímni používatelia aplikácie Handyman nebudú ovplyvnení potenciálnymi útokmi a budú môcť aplikáciu používať bez obmedzení. [39] [40]

```
const limiter : RateLimitRequestHandler = rateLimit( passedOptions: {
  windowMs: 5 * 60 * 1000, // 5 minutes
  max: 100, // Limit each IP to 100 requests per `window` (here, per 5 minutes
  standardHeaders: true,
  legacyHeaders: false,
})
```

Obrázok 28. Request limiter konfigurácia

5.3 Helmet balíček

Pre zabezpečenie webovej aplikácie Handyman a ochranu proti mnohým bežným zraniteľnostiam webových aplikácií, bola implementovaná knižnica Helmet. Helmet je middleware pre Express.js, ktorý nastavuje bezpečnostné HTTP hlavičky. Tieto hlavičky hrajú kľúčovú rolu v ochrane pred rôznymi druhmi útokov, ako sú cross-site scripting (XSS), clickjacking a iné. [41]

Helmet je vlastne súbor menších middleware funkcií, ktoré nastavujú konkrétne HTTP hlavičky. Niektoré z týchto hlavičiek zahŕňajú Strict-Transport-Security, X-Content-Type

Options, X-Frame-Options a X-XSS-Protection. Každá z týchto hlavičiek zabezpečuje ochranu proti konkrétnemu druhu útoku. [41]

Použitím Helmet balíčka sa zvyšuje bezpečnosť webovej aplikácie Handyman tým, že sa zabraňuje preniknutiu nebezpečných útokov prostredníctvom konfigurácie bezpečnostných HTTP hlavičiek. Tento balíček pomáha udržať bezpečnosť aplikácie na vysokej úrovni, a tak zabezpečuje, že užívatelia môžu bezpečne používať aplikáciu bez obáv z potenciálnych kybernetických útokov. [41]

5.4 Implementácia CORS politiky

Ako súčasť zabezpečenia webovej aplikácie Handyman bola prijatá a implementovaná politika Cross-Origin Resource Sharing (CORS). CORS je štandardný mechanizmus, ktorý umožňuje alebo zakazuje zdieľanie zdrojov medzi webovými stránkami s rôznymi pôvodmi. CORS politika v aplikácii Handyman je nastavená tak, že kontroluje, z ktorých pôvodov môže aplikácia prijímať požiadavky, a tak poskytuje ďalšiu vrstvu ochrany pred potenciálnymi útokmi. Implementáciou CORS politiky sa tak zaisťuje, že len overené zdroje môžu interagovať s aplikáciou Handyman.

6 ZHRNUTIE IMPLEMENTÁCIE A POROVNANIE S OSTATNÝMI TECHNOLÓGIAMI

Táto sekcia je venovaná zhrnutiu procesu implementácie webovej aplikácie Handyman a porovnaniu použitých technológií s alternatívami, ktoré mohli byť potenciálne použité v priebehu vývoja aplikácie.

6.1 Zhrnutie implementácie

Implementácia webovej aplikácie Handyman bola uskutočnená pomocou MERN stacku, ktorý zahŕňa MongoDB, Express.js, React.js a Node.js.

Pomocou MongoDB bola vytvorená databáza, do ktorej sa ukladali dáta o užívateľoch, zákaziek atď. Výhodou využitia tejto NoSQL databázy je jednoduchá integrácia s Node.js a Express.js a taktiež pre jej škálovateľnosť a používateľskej prívetivosti.

Node.js a Express.js vytvárajú základ serverovej časti aplikácie. Pomocou nich bola vytvorená API, ktorá slúži na komunikáciu medzi serverom a klientom.

React.js: Na front-end strane, React.js umožnil vytvorenie komponentov, formulárov a celého užívateľského rozhrania.

6.2 Porovnanie použitých technológií s alternatívami

Táto podkapitola je venovaná porovnávaniu použitých technológií s vhodnými alternatívami. K alternatívam sú popísané ich základné vlastnosti a charakteristiky. Ďalej sú v rámci porovnania uvedené výhody a nevýhody daných alternatív.

6.2.1 MERN stack vs MEAN stack

Porovnanie MERN stacku a MEAN stacku zahŕňa rozdiely vo front-end technológiách, konkrétne využitie React.js v MERN stacku a Angular v MEAN stacku. Nasledujúce rozdiely boli identifikované:

React.js vs Angular:

- React.js je jednoduchší na naučenie sa v porovnaní s Angularom, čo môže uľahčiť vstup do vývoja pre nových programátorov. [42]
- React.js má rýchlejšiu odozvu v UI vykresľovaní, pretože využíva Virtual DOM. Naopak, Angular má mierne nižší výkon pri vykresľovaní UI. [42]

- React.js je preferovaný pri tvorbe menších aplikácií alebo jednostránkových aplikácií (SPA), zatiaľ čo Angular je častejšie využívaný pri tvorbe väčších aplikácií. [42]
- React.js používa jednosmerný "data binding" pre aktualizáciu stavu a zobrazenie dát, zatiaľ čo Angular využíva obojsmerný "data binding". [42]
- React.js má vlastný framework pre vytváranie mobilných aplikácií, zatiaľ čo Angular nemá vlastný nástroj pre vytváranie natívnych aplikácií. [42]

Na základe tohto porovnania je zrejmé, že MERN stack je vhodnejšou voľbou pre aplikáciu Handyman. MERN stack ponúka jednoduchosť, rýchlu odozvu v UI prostredníctvom Virtual DOM, čo vedie k lepšiemu užívateľskému zážitku (UX). Taktiež je MERN stack vhodný pre aplikáciu Handyman ako SPA (Single Page Application). [42]

6.2.2 MySQL vs MongoDB

MySQL je tradičný relačný databázový systém založený na štruktúrovanom dotazovacom jazyku SQL. MongoDB je NoSQL databázový systém založený na BSON dokumentoch a schemaflexibilných databázach. Porovnanie je nasledovné:

Tabuľka 1. Porovnanie: MySQL – MongoDB [43]

	MySQL	MongoDB
Model dát	Tabuľkový model dát	Key-Value model dát
Schéma	Nutnosť dodržiavať pred-definované schéma	Nie je nutnosť mať predde-finované pevné schéma
Jazyk na manipuláciu s dátami	SQL	Vlastný dotazovací jazyk
Výkonnosť a škálovateľnosť	Rýchlejšia pri menších dátových sadách	Výkonnejšia pri väčších dátových sadách
Transakcie	Podporuje plné transakcie so všetkými vlastnosťami ACID	Podpora pre multi-dokumentové transakcie
Spoľahlivosť a dostupnosť	Master-Slave replikačný model	Model replikácie založený na replikačných súpravách

Pre aplikáciu Handyman bola použitá databáza MongoDB kvôli tomu, že to umožnilo efektívnejšiu implementáciu pomocou knižnice mongoose, ktorá sa dá nainšalovať pomocou NPM balíčka a umožňuje jednoduchú prácu s databázou bez nutnosti pokročilých znalostí MongoDB. A však je nutné dodať, že MySQL by bola vhodná kvôli jej podpore transakcií s vlastnosťami ACID, ale výhody MongoDB a jej použitia spolu s Node.js a Express.js prevažujú tento nedostatok.

ZÁVER

V závere tejto bakalárskej práce by mal bežný čitateľ mať jasný obraz o procese vývoja webovej aplikácie. Prvotná časť práce poskytla východiskový bod, kde boli predstavené základné koncepty a terminológie, ktoré sa neskôr uplatnili pri praktickej implementácii.

Čitateľ bol detailne informovaný o existujúcej konkurencii na trhu a o možnostiach, ktoré sa naskytujú pri využití nedostatkov týchto konkurenčných produktov pre vytvorenie novej, zlepšenej platformy pre zákazníkov a poskytovateľov služieb.

Boli jasne definované funkčné a nefunkčné požiadavky, ktoré určili očakávaný výkon a ciele webovej aplikácie. Ďalej bol vytvorený a prezentovaný dizajn webovej aplikácie, ktorý slúžil ako vizuálny základ pre jej ďalší vývoj.

Pri tvorbe webovej aplikácie bola prioritou implementácia základných funkcionalít, vrátane vytvorenia nového účtu, prihlásenia, vytvorenia a prezerania zákaziek, filtrovania zákaziek, autorizácie a autentifikácie pomocou JWT a zabezpečenia aplikácie proti potenciálnym kybernetickým hrozbám.

Všetky tieto kroky viedli k úspešnému vytvoreniu základnej verzie webovej aplikácie Handyman. Tento prototyp môže poslúžiť ako štartovací bod pre ďalší vývoj a potenciálne nasadenie tejto aplikácie do reálneho prostredia.

Cieľom tejto bakalárskej práce bolo poskytnúť jasný a detailný prehľad o procese vývoja webovej aplikácie od konceptu až po implementáciu. V zmysle týchto cieľov bola práca úspešne dokončená a dúfam, že poskytne hodnotný základ pre ďalšie vývojové práce a výskum v tejto oblasti.

ZOZNAM POUŽITEJ LITERATÚRY

[1] Getting started with the web [online]. [cit. 2023-04-25]. Dostupné z: https://developer.mozilla.org/enUS/docs/Learn/Getting_started_with_the_web/HTML_basics

[2] B., Artūras. What Is CSS and How Does It Work? [online]. 04.01.2023, 1 [cit. 2023-04-25]. Dostupné z: https://www.hostinger.com/tutorials/what-is-css#What_Is_CSS

[3] PARUCH, Zach. What Is JavaScript & What Is It Used For? A Basic Guide to JS [online]. Mar 21, 2023, 1 [cit. 2023-04-25]. Dostupné z: https://www.semrush.com/blog/javascript/?kw=&cmp=EE_SRCH_DSA_Blog_EN&label=dsa_pagefeed&Network=g&Device=c&kwid=dsa1753200738893&cmpid=18361923498&agpid=140825965145&BU=Core&extid=60162843321&adpos=

[4] 2022 Developer Survey [online]. [cit. 2023-04-25]. Dostupné z: https://survey.stackoverflow.co/2022/#most-popular-technologies-language

[5] GRAVELLE, Robert. Top JavaScript Frameworks [online]. [cit. 2023-04-25]. Dostupné z: https://www.developer.com/languages/javascript/top-javascript-frameworks/

[6] SYED, Ateeq. Why you should learn TypeScript in 2023? [online]. [cit. 2023-04-25]. Dostupné z: https://dev.to/syedateeq160/why-you-should-learn-typescript-in-2023-2hgn#:~:text=TypeScript%20is%20a%20powerful%20and,and%20work%20on%20exciting%20projects.

[7] MINNICK, Chris. Beginning React JS Foundations building user interfaces with ReactJS. 1. Canada: John Wiley, 2022. ISBN 978-1-119-68554-8.

[8] Thinking in React. React.dev [online]. [cit. 2023-04-26]. Dostupné z: https://react.dev/learn/thinking-in-react

[9] CHAVAN, Yogesh. JSX in React – Explained with Examples. Freecodecamp [online]. 2021 [cit. 2023-04-26]. Dostupné z: https://www.freecodecamp.org/news/jsx-in-react-introduction/

[10] ARANCIO, Stephen. What is JSX?. Medium [online]. 2021 [cit. 2023-04-26]. Dostupné z: https://medium.com/@sjarancio/what-is-jsx-e3dda0af3490

[11] Understand all about Props In React Js. Simplilearn [online]. 2023 [cit. 2023-04-26]. Dostupné z: https://www.simplilearn.com/what-is-reactjs-props-article

[12] React Props Explained with Examples. Refine [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://refine.dev/blog/react-props/

[13] React Hooks Fundamentals for Beginners. Freecodecamp [online]. 2022 [cit. 2023-04-26]. Dostupné z: https://www.freecodecamp.org/news/react-hooks-fundamentals/

[14] Built-in React Hooks. React [online]. 2023 [cit. 2023-04-26]. Dostupné z: https://react.dev/reference/react

[15] Ultimate React Router v6 Guide. Webdevsimplified [online]. 2022 [cit. 2023-05-09]. Dostupné z: https://blog.webdevsimplified.com/2022-07/react-router/

[16] What is Node.js: A Comprehensive Guide. Simplilearn [online]. 2023 [cit. 2023-05-09]. Dostupné z: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

[17] What is Node JS Used For in 2023 – Complete Guide. Jaydevs [online]. 2023 [cit. 2023-05-09]. Dostupné z: https://jaydevs.com/what-is-node-js-used-for/

[18] Features of Node.js. Educba [online]. [cit. 2023-05-09]. Dostupné z: https://www.educba.com/features-of-node-js/

[19] What is NPM? A Beginner's Guide. Careerfoundry [online]. 2022 [cit. 2023-05-09]. Dostupné z: https://careerfoundry.com/en/blog/web-development/what-is-npm/#what-is-npm

[20] What Is Express.js? Everything You Should Know. Kinsta [online]. 2022 [cit. 2023-05-09]. Dostupné z: https://kinsta.com/knowledgebase/what-is-express-js/

[21] What is Express.js?. Codecademy [online]. c2023 [cit. 2023-05-09]. Dostupné z: https://www.codecademy.com/article/what-is-express-js

[22] MongoDB. Techtarget [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[23] Introduction to Mongoose for MongoDB. Freecodecamp [online]. 2018 [cit. 2023-05-10]. Dostupné z: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/

[24] Guides. Mongoosejs [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://mongoosejs.com/docs/guides.html

[25] What is MERN stack and how does it work?. Imaginarycloud [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://www.imaginarycloud.com/blog/what-is-mern-stack-and-how-does-it-work/

[26] The Power of Figma as a Design Tool. Toptal [online]. 2018 [cit. 2023-05-11]. Dostupné z: https://www.toptal.com/designers/ui/figma-design-tool

[27] What Is Figma and What Is It Used For?. Makeuseof [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.makeuseof.com/what-is-figma-used-for/

[28] How To Choose An Eye-Catching Website Color Palette. Websitebuilderexpert [online]. 2023 [cit. 2023-05-16]. Dostupné z: https://www.websitebuilderexpert.com/designing-websites/how-to-choose-color-for-your-website/

[29] Basic Types of Buttons in User Interfaces. Uxplanet [online]. 2019 [cit. 2023-05-16]. Dostupné z: https://uxplanet.org/basic-types-of-buttons-in-user-interfaces-ea7b065f66ee

[30] Choosing web fonts beginners guide. Google [online]. 2018 [cit. 2023-05-16]. Dostupné z: https://design.google/library/choosing-web-fonts-beginners-guide

[31] 20 best fonts for websites in 2022. Editorx [online]. 2021 [cit. 2023-05-16]. Dostupné z: https://www.editorx.com/shaping-design/article/best-fonts-for-websites

[32] Interfaces. Typescriptlang [online]. 2023 [cit. 2023-05-21]. Dostupné z: https://www.typescriptlang.org/docs/handbook/interfaces.html

[33] Express Tutorial Part 4: Routes and controllers. Mdn web docs [online]. 2023 [cit. 2023-05-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

[34] What is Adobe XD and what is it used for?. Adobe [online]. 2023 [cit. 2023-05-24]. Dostupné z: https://www.adobe.com/cz/products/xd/learn/get-started/what-is-adobe-xd-used-for.html

[35] What Is Sketch and What Can You Do With It?. Makeuseof [online]. 2022 [cit. 2023-05-24]. Dostupné z: https://www.makeuseof.com/what-is-sketch/

[36] Framer vs Figma: Which is a Better Design Tool?. Pttrns [online]. 2022 [cit. 2023-05-24]. Dostupné z: https://www.pttrns.com/framer-vs-figma-which-is-a-better-design-tool/

[37] JSON Web Tokens. Auth0: Secure access for everyone. But not just anyone. [online]. Copyright © [cit. 23.05.2023]. Dostupné z: https://auth0.com/docs/secure/tokens/json-web-tokens

[38] Httponly Cookie Javascript: Javascript Explained – Bito. Bito [online]. Copyright © 2023 [cit. 23.05.2023]. Dostupné z: https://bito.ai/resources/httponly-cookie-javascript-javascript-explained/#:~:text=Benefits%20of%20Using%20Httponly%20Cookies&text=Httponly%20Cookies%20also%20provide%20an,from%20being%20stolen%20or%20manipulated.

[39] express-rate-limit - npm. npm [online]. Copyright © [cit. 23.05.2023]. Dostupné z: https://www.npmjs.com/package/express-rate-limit

[40] What is Express-rate-limit in Node.js ? - GeeksforGeeks. GeeksforGeeks | A computer science portal for geeks [online]. Dostupné z: https://www.geeksforgeeks.org/what-is-express-rate-limit-in-node-js/

[41] Using Helmet in Node.js to secure your application - LogRocket Blog. LogRocket Blog - Resources to Help Product Teams Ship Amazing Digital Experiences [online]. Dostupné z: https://blog.logrocket.com/using-helmet-node-js-secure-application/#:~:text=js!-,What%20is%20Helmet%3F,comply%20with%20web%20security%20standards.

[42] MEAN vs MERN - The Ultimate Comparison 2023 | Clean Commit. Application Development & Headless eCommerce | Clean Commit [online]. Copyright © 2018 [cit. 24.05.2023]. Dostupné z: https://cleancommit.io/blog/mean-vs-mern-the-ultimate-comparison-2022/

[43] What is the Difference between MongoDB and SQL?. Scaler [online]. 2023 [cit. 2023-05-25]. Dostupné z: https://www.scaler.com/topics/mongodb-vs-sql/

ZOZNAM POUŽITÝCH SKRATIEK A SYMBOLOV

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
Atď	A tak ďalej
BSON	Binary JavaScript Object Notation
CRUD	Create Read Update Delete
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
DDoS	Distributed Denial-of-Service
DOM	Document Object Model
Db	Database
HTML	Hypertext Markup Language
http	Hypertext Transfer Protocol
I/O	Input/Output
IP	Internet Protocol
Id	Identification
JS	JavaScript
JSX	JavaScript XML
JWT	JSON Web Token
JSON	JavaScript Object Notation
NavBar	Navigation Bar
MEAN	Mongo Express.js Angular Node.js
MERN	Mongo Express.js React.js Node.js
NPM	Node Package Manager
NoSQL	Non Structured Query Language
ODM	Object Data Modeling

Props	Properties
SQL	Structured Query Language
SPA	Single Page Application
TS	TypeScript
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language
XSS	Cross-site Scripting

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha P I: CD

PRÍLOHA P I: CD

Obsahom CD je zdrojový kód, prihlasovacie údaje a obrázky dizajnu vo figme.