SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE
dokumentace

Planeta kupónů

Jan Čech

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2021/2022

Poděkování

V úvodu této práce bych chtěl poděkovat panu učiteli Mgr. Marku Lučnému a panu učiteli Ing. Petru Grussmannovi za vstřícnost a ochotu pomoci.

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2021

podpis autora práce

ANOTACE

Výsledkem práce je částečně zautomatizovaná webová aplikace, která uživatelům zprostředkovává kupóny pro určité internetové obchody.

Frontend aplikace stojí na velice rychlé javascriptové knihovně React 17.0.2, která je optimální pro práci s rychle se měnícími daty, v tomto případě s obchody a kupóny.

Backend aplikace je tvořen ve frameworku Flask 2.0.1, který je schopen vytvořit datový model a dělat zápisy do databáze. Rozšíření Flask Admin 1.5.8 dodává možnost vytvořit si vlastní, uživatelsky přívětivé, administrační prostředí.

Webová aplikace je zcela zdokerizována, tudíž její spuštění vyžaduje použití jednoho příkazu.

Na tvorbě se podíleli tři lidé, jmenovitě: Jan Čech, Matěj Čech a Lukáš Rychlý.

OBSAH

ÚVOD **6**

1 WEBOVÁ APLIKACE, FRONTEND **7**

1.1 FRAMEWORK 7

1.2 WEBOVÁ APLIKACE 7

1.3 JAK FUNGUJE WEBOVÁ APLIKACE 7

1.3.1 Webový server 7

1.3.2 Statická stránka 8

1.3.3 Dynamická stránka 8

1.3.4 Vícestránková webová aplikace 9

1.3.5 Vícestránková webová aplikace (React) 10

1.4 DOM X VDOM (REACT) 10

1.5 KOMPONENTY 11

1.5.1 Komponenty v Reactu 11

1.6 HOOK (REACT) 11

1.7 ZÍSKÁVÁNÍ DAT Z API 11

2 VYUŽITÉ TECHNOLOGIE **13**

2.1 FIGMA 13

2.2 REPLIT 13

2.3 CLOCKIFY 14

2.4 REACT.JS 14

2.5 FLASK 15

2.6 POSTGRESQL 15

2.7 DOCKER & DOCKER COMPOSE 15

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY **17**

3.1 KOMPONENTY 17

3.2 HOOK (REACT) 19

3.1 REAKTIVNÍ PROMĚNNÉ 20

3.2 SINGLE-PAGE A MULTI-PAGE V REACTU 20

4 VÝSLEDKY ŘEŠENÍ, UŽIVATELSKÝ MANUÁL **22**

4.1 CÍLE PROJEKTU 22

4.2 UŽIVATELSKÝ MANUÁL 22

4.2.1 Běžný uživatel 22

4.2.2 Uživatel s administrátorskými právy 23

ZÁVĚR **24**

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ 25

ÚVOD

Cílem naší práce bylo vytvořit webovou aplikaci, která bude uživatelům internetu zprostředkovávat slevové kupóny z různých online obchodů.

Proč jsme si zvolili právě toto téma?

Během první vlny koronavirové pandemie byla valná většina z nás nucena používat počítač víc, než jsme byli zvyklí. Jelikož nákup v kamenných obchodech představoval jisté riziko, našel český národ prahnoucí po slevách útočiště v online světě. Díky této situaci nás napadlo vytvořit webové prostředí, kde by zmínění uživatelé nalezli přesně to, co hledají – všechny kupóny na jednom místě.

Automatizace

Web jsme se snažili co nejvíce automatizovat, tak, aby byl zásah správců co nejmenší a byl schopen nezávislé existence.

Postup

Tato dokumentace popisuje naši cestu ke konečnému řešení – od nápadu po realizaci. Dále zahrnuje výpis použitých technologií, způsoby řešení a jejich konkrétní příklady, sebehodnocení a naši vizi projektu v budoucnosti.

1 WEBOVÁ APLIKACE, FRONTEND

Frontend (nebo také *front end*) je grafická část systému, jež je viditelná pro všechny uživatele a kterou přímo ovládá uživatel.

1.1 Framework

Framework slouží jako podpora při programování a vývoji softwarových projektů. Úkolem frameworku je převzít časté problémy dané oblasti, čímž se usnadní vývoj – vývojáři se mohou pouze soustředit na své zadání.

Příkladem frameworku je např. Angular, vyvíjený firmou Google. Angular je robustní open-source frontendový framework založený na TypeScriptu Kvůli velké obsáhlosti a robustnosti Angularu jsme si na tento projekt pro naše účely vybrali velice dostačující React. Nejedná se však o framework, React je JavaScriptová knihovna pro tvorbu uživatelského rozhraní.

1.2 Webová aplikace

Webová aplikace je webové místo, které obsahuje stránky s úplně nebo částečně určitým obsahem. Tento obsah se načte v tu chvíli, kdy uživatel požádá o stránku ze serveru. Konečný obsah se liší podle požadavku a závisí na akcích uživatele, takovéto stránce se říká dynamická.

1.3 Jak funguje webová aplikace

Webová aplikace je shromáždění statických a dynamických stránek. Tyto stránky jsou poskytovány přímo ze serveru přes internet. Obvykle funguje pro více platforem, takže je schopná pracovat bez ohledu na operačním systému koncového uživatele.

1.3.1 Webový server

Webový server je program, který je odpovědný za vyřizování požadavků HTTP od klientů (nejčastěji webových prohlížečů). Na základě požadavku zasílá klientům webové stránky.

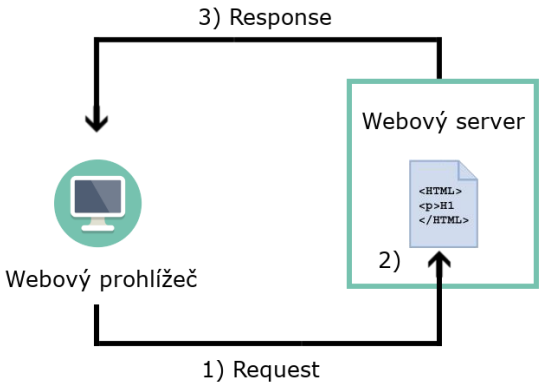
1.3.2 Statická stránka

Statická stránka se nijak nemodifikuje. Když o ni návštěvník požádá, je mu webovým serverem odeslána stránka pro webový prohlížeč, který o ni požádal, beze změny, je doručena uživateli přesně tak, jak je uložena na webovém serveru.

Statická stránka nemusí být striktně statická, může obsahovat např. různé přechody, efekty. Stále se však jedná o statickou stránku, jelikož nedochází k modifikacím a změnám před odesláním.

Zpracování statické webové stránky:

Statické webové místo obsahuje HTML soubory a soubory s tím spojené (např. CSS, mediální soubory, …) hostované na webovém serveru.



Obrázek 1.3.2: Zpracování statické stránky

Vysvětlení obrázku č. 1.3.2: 1) Webový prohlížeč (klient) požádá o statickou stránku. 2) Webový server tuto stránku vyhledá. 3) Webový server zašle stránku prohlížeči.

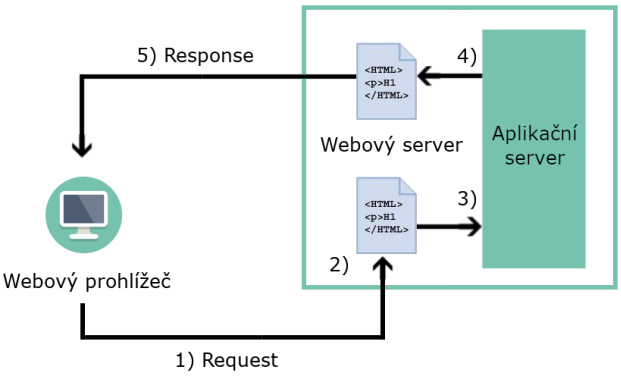
1.3.3 Dynamická stránka

Jak název napovídá, pro dynamické stránky je charakteristický neustále se měnící obsah. Dynamická stránka je serverem modifikována před odesláním prohlížeči, který o ni požádal.

Zpracování dynamické webové stránky:

Když webový server dostane požadavek na odeslání statické webové stránky, tak stránku odešle přímo klientovi. Pokud se jedná o stránku dynamickou, reaguje webový server tro-

chu jinak: předá stránku speciálnímu softwaru (aplikační server), aby dokončil stránku. Aplikační server dokončí stránku a jako výsledek dorazí webovému prohlížeči stránka statická, tedy čisté HTML.

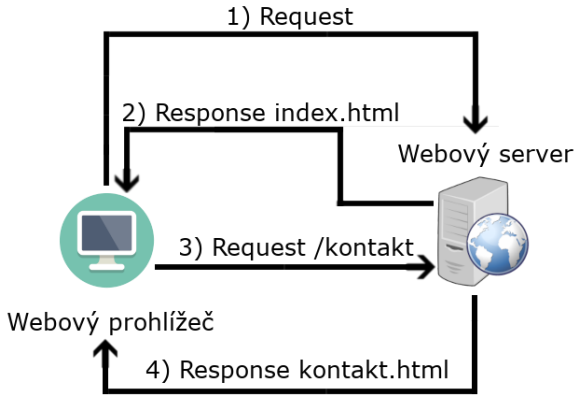


Obrázek 1.3.3: Zpracování dynamické stránky

Vysvětlení obrázku č. 1.3.3: 1) Webový prohlížeč (klient) požádá o dynamickou stránku. 2) Webový server tuto stránku vyhledá a předá ji aplikačnímu serveru. 3) Aplikační server stránku dokončí. 4) Aplikační server předá dokončenou stránku webovému serveru. 5) Webový server zašle stránku prohlížeči.

1.3.4 Vícestránková webová aplikace

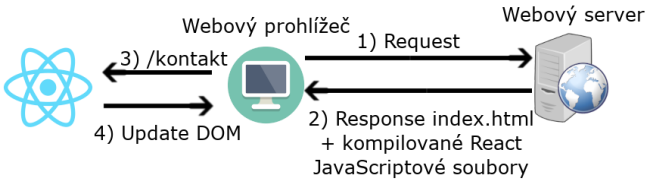
U vícestránkového webu se při každém prokliku na jinou stránku zasílá požadavek na webový server, ten jako odpověď zasílá statickou stránku zpět.



Obrázek 1.3.4: Vícestránkový web

1.3.5 Vícestránková webová aplikace (React)

U Reactu je to s vícestránkovými weby jinak, měnění obsahu je ponecháno pouze na webovém prohlížeči. Při prvním requestu na webový server se jako response vrátí HTML soubor a k tomu kompilované React JavaScriptové soubory, které ovládají React aplikaci. Takže React a React Router mají plnou kontrolu nad aplikací, tzn. pokud bychom klikli na odkaz, který nás má přesměrovat na jinou podstránku, tak React Router do tohoto procesu vstoupí a „odrazí“ další požadavek na server, podívá se na tento požadavek a následně vloží požadovaný kontent na stránku.



Obrázek 1.3.5: Vícestránkový web (React)

1.4 DOM x VDOM (React)

DOM (Document Object Model) je datová reprezentace struktury webové stránky (UI). Manipulace s DOM je velice pomalá, např. pokud je změněná některá část stránky (klidně i malá část), tak se musí celé UI znovu překreslit.

Virtual DOM (VDOM) tvoří kopii originálního DOM uloženou v paměti. VDOM je synchronizovaný s reálným DOM pomocí knihovny jako je ReactDOM. Tento proces se nazývá anglicky Reconciliation (= proces zajišťování shody dvou sad záznamů).

VDOM na rozdíl od DOM není schopen přímo měnit obsah na obrazovce. Když je zaznamenána změna ve vitual DOM, React porovná VDOM z tohoto okamžiku se snapshotem VDOM právě před změnou. Pomocí tohoto porovnání React zjistí, které části se změnily a které je nutné znovu vykreslit. Ve chvíli, kdy React ví, jaká část je změněna, je nahrazena jen a pouze tato část v originálním DOM.

1.5 Komponenty

Komponenty představují stavební bloky, ze kterých se skládá výsledný vzhled. Jejich vícenásobné použití odstraňuje redundantní části kódu. Komponenty je možné vnořovat do sebe, a tím vytvářet jednotlivé bloky, stránky či podstránky.

1.5.1 Komponenty v Reactu

Komponenty jsou JavaScriptové soubory, které přebírají libovolné vstupy zvané props a vracejí React elementy, které popisují, co by se mělo zobrazovat na obrazovce.

V Reactu jsou dva způsoby, jak zapisovat komponenty a těmi jsou: komponenty tvořené z funkcí a komponenty tvořené ze třídy.

Příklad kódu vytvoření Function Component:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Příklad kódu vytvoření Class Component:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Obě ukázky komponentů dosáhnou téhož cíle, pouze jsou jinak zapsány.

1.6 Hook (React)

React Hooks (představeny v Reactu ve verzi 16.8) jsou JavaScriptové funkce, které umožňují stavět komponenty Reactu pouze funkcemi (Function Components). React poskytuje několik Hooků pro správu většiny případů, ale také přichází se způsobem, jak si vytvořit vlastní Hooky. Název Hooku začíná slovem *use* např. `useEffect, useState`, …

1.7 Získávání dat z API

Pro naplnění stránky daty, které nejsou určité, stále se mění a jsou uložena v databázi, je třeba použít nějaké metody, kterou tyto data získáme.

V JavaScriptu existuje metoda `fetch()`, která je dostupná v global scope a vrací tzv. Promise. Fetch api interface dovoluje webovým prohlížečům dělat tzv. HTTP requesty na webové servery.

2 VYUŽITÉ TECHNOLOGIE

Námi vytvořená webová aplikace je napsána zejména v jazycích JavaScript, Python 3.8, HTML5, CSS3. Pro databázi jsme zvolili PostgreSQL 13.4.

2.1 Figma

Figma je vektorový grafický editor pro vytváření prototypů. Využili jsme jej k převedení našich představ do reálné podoby. Jedná se o velmi intuitivní editor, který umožňuje jednoduchou práci s objekty na virtuálním plátně. Na návrhu může v reálném čase spolupracovat více lidí.

Obrázek 2.1: Figma

2.2 Replit

Replit představuje online vývojové prostředí, které umožňuje zaregistrovaným uživatelům vytvářet aplikace a webové stránky pomocí prohlížeče. Tato platforma nabízí různé funkce pro spolupráci, včetně možnosti úprav pro více uživatelů v reálném čase. Kód je možné spustit přímo v prohlížeči. Tuto službu jsme využili při vytváření komponentů v React.js.

Obrázek 2.2: Replit

2.3 Clockify

Clockify je online aplikace sloužící k sledování času a vytváření časového rozvrhu. Použili jsme ji jako záznamník námi vykonané práce. Na základě statistik stráveného času automaticky převádí zaznamenané hodnoty do přehledných grafů.

Obrázek 2.3: Clockify

2.4 React.js

React je JavaScriptová knihovna, která slouží k tvorbě uživatelského rozhraní. Je optimální pro práci s rychle se měnícími daty. Specifická syntaxe JSX neboli JavaScript XML má podobný vzhled jako HTML, značně tak zjednodušuje vytváření komponentů programátorům, kteří se již setkali s klasickým HTML. Námi použitá verze: 17.0.2.

Obrázek 2.4: React.js

2.5 Flask

Flask je webový mikro framework napsaný v programovacím jazyce Python. Na rozdíl od ostatních frameworků (například Django) tento mikro framework nenabízí žádné administrační rozhraní, podporuje však rozšíření, která mohou přidávat do aplikace další funkce, jako by byly implementovány v samotném Flasku. Jelikož pro naši aplikaci nepotřebujeme robustní framework, na základě doporučení jsme si zvolili právě Flask 2.0.1.

Obrázek 2.5: Flask

2.6 PostgreSQL

Jedná se o open-source objektově-relační databázový systém, tato platforma je robustní a zároveň bezpečná. Databázi je možné spojit s Flaskem pomocí knihovny SQL-Alchemy a umožnit tak jednoduchou práci s modely.

Obrázek 2.6: PostgreSQL

2.7 Docker & Docker Compose

Docker je open-source software, který slouží k izolaci aplikací do kontejnerů. Takto vytvořený kontejner obsahuje pouze požadované aplikace a s nimi spjaté soubory, nikoliv však operační systém.

Docker Compose je nástroj pro spouštění více kontejnerů najednou. Tento nástroj jsme tedy využili pro hromadné spuštění jednotlivých částí našeho projektu jedním příkazem (backend, frontend, databáze).

Obrázek 2.6: Docker

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 Komponenty

Komponenty jsou v Reactu JavaScriptové třídy nebo funkce, které se skládají ze dvou hlavních částí, logiky komponentu a jeho vykreslení, které se vrací returnem v podobě JSX. Obsah returnu se až na pár odlišností dosti podobá klasickému HTML.

Příklad rozdělenosti komponentu *Search.jsx*:

Obrázek 3.1: Vyhledávání

Logika komponentu:

```
const url = '/api/vypis-obchodu';
const {data: everyShop} = useFetch(url);
const [searchTerm, setSearchTerm] = useState('');
const [shops, setShops] = useState(null);

useEffect(() => {
  // Něco je zadáno
  if (searchTerm !== "") {
    setShops(everyShop.filter(shop => {
      if (shop.nazev.toLowerCase().includes(searchTerm.toLowerCase())) {
        return shop;
      }
      else {
        return "";
      }
    }))
  }
  // Pokud není nic ve vyhledávači
  else {
    setShops(null);
  }
}, [searchTerm]);

const handleChange = (val) => {
  setSearchTerm(val);
}
```

Obsah returnu (pouze část pro lepší orientaci):

```
<input type="search" className="form-control rounded" placeholder="Vyhledat
obchod..." aria-label="Search" aria-describedby="search-addon"
onChange={e => { handleChange(e.target.value) }} />

{shops && <ul className="search-suggestions">{shops.slice(0, 5).map(shop =>
(<Link to={"/obchod/" + shop.id} key={shop.id} >
<li>{shop.nazev}</li></Link>))}</ul>}
```

Logika komponentu kontroluje, zdali byl zadán nějaký řetězec do vyhledávacího pole, pokud ano, porovná se tento řetězec se všemi obchody, které jsou v databázi uloženy. Pokud se string shoduje s nějakým obchodem, je vykreslen jako výsledek pod vyhledávací pole. V případě, že je výsledků více, zobrazí se jich nanejvýš pět.

Z komponentů lze pak skládat celé stránky. Na rozdíl od čistého HTML, musí všechny značky být párové, i ty, co v HTML párové nejsou, anebo rovnou ukončeny.

Ukázka skládání stránky z komponentů:

```
<>
  <Mezera mezera={2} />
  <HeaderUvod />
  <Mezera mezera={5} />
  <BannerUvod />
  <Mezera mezera={5} />
  <KategorieNadpis />
  <Mezera mezera={4} />
  <Container className="mt-4">
      <Row>
        {kategorie.map((kat, i) => {<KategorieUvod key={i}
        nadpis={kat.nadpis} pozadi={kat.pozadi}
        text={kat.text} odkaz={kat.odkaz} />})}
      </Row>
  </Container>
</>
```

Jak je z ukázky patrné, skládání komponentů je dosti podobné HTML. Na rozdíl od HTML je zde použité slovo `className` namísto `class`, jelikož se jedná o JavaScript, kde slovíčko class už je obsaženo.

3.2 Hook (React)

V Reactu je od verze 16.8 možnost používat předdefinované Hooky, ale také vytvářet vlastní. Vlastní Hooky mohou pomoci například se získáváním (fetchováním) dat a odstraní tak repetitivní části kódu.

Ukázka vlastního Hooku *useFetch.js*:

```
const useFetch = (url) => {
    const [data, setData] = useState(null);
    const [isPending, setIsPending] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        async function fetchData() {
            const abortCont = new AbortController();

            try {
                const res = await fetch(url, { signal: abortCont.signal }, {
                    headers: { "Content-Type": "application/json",
                    "Charset": "utf-8" },
                });
                if (!res.ok) {
                    throw Error("Could not fetch the data \
                    for that resource");
                }
                const data = await res.json();
                setData(data);
                setIsPending(false);
                setError(null);
            }
            catch (err) {
                if (err.name === "AbortError") {
                    console.log("fetch aborted");
                }
                else {
                    setIsPending(false);
                    setError(err.message);
                }
            }
            return () => abortCont.abort();
        }
        fetchData();
    }, [url]);
    return { data, isPending, error };
}
```

Tento Hook přijímá jako parametr url adresu, ze které se pokusí získat data a vrací nám jak data v JSON formátu (za podmínky, že se je povedlo získat), tak informaci o tom, zdali se stále čeká na získání dat, anebo zdali došlo k chybě.

Pokud komponent, který se snažíme updatnout, v danou chvíli není přítomen na stránce (je unmounted), je fetch zastaven AbortControllerem.

3.1 Reaktivní proměnné

V Reactu pokud změníme hodnotu proměnné, kterou vykreslujeme na stránce, tak nedosáhneme překreslení na novou hodnotu (hodnota sama o sobě se změní, jen nedojde k jejímu překreslení ve vyobrazení), jelikož React nesleduje tyto změny, tudíž tato proměna není reaktivní. Abychom dosáhli překreslení musíme použít ve funkcionálních komponentech Hook zvaný `useState`, tento Hook se o změnu, překreslení, postará.

Využití reaktivní proměnné ve vyhledávači:

```
const [searchTerm, setSearchTerm] = useState('');

const handleChange = (val) => {
  setSearchTerm(val);
}

return (
    <>
      <input type="search" className="form-control rounded"
      placeholder="Vyhledat obchod..." aria-label="Search"
      aria-describedby="search-addon"
      onChange={e => { handleChange(e.target.value) }} />

      {searchTerm}
    </>
)
```

3.2 Single-Page a Multi-Page v Reactu

React může být využit jako základ pro tvorbu single-page. Jak ale v Reactu vytvořit vícestránkový web?

V základu v Reactu není přímo nějaká možnost, jak vytvořit vícestránkovou aplikaci, proto na scénu vstupuje balíček React Router. Ten nám dovolí roztřídit si stránku tak, abychom překreslovali jen ty části, které se liší od ostatních stránek (tzn. že například navbar dáme

jako společný prvek všech stránek, ale obsah, body, bude záviset na stránce, na které se právě nacházíme).

Ukázka rozčlenění prvků v závislosti na stránce:

```
<Router>
  <NavbarComponent />
  <Switch>
    <Route exact path="/">
      <Uvod />
    </Route>
    <Route path="/seznam">
      <Seznam />
    </Route>
    <Route exact path="/obchod/:id">
      <Obchod/>
    </Route>
    <Route component={NotFound} />
  </Switch>
</Router>
```

NavbarComponent je prvek společný. To, co je uvnitř Switche je obsah proměnlivý, v závislosti na cestě (path).

Abychom se vyhnuli stálému posílání požadavků na server, jak je uvedeno v kapitole 1.3.4. a 1.3.5., je třeba v odkazech místo tagu `<a>` použít značky `Link`, která je taktéž obsažena v balíčku React Router. Ve vykreslení na obrazovku je sice Link přetvořen do anchor tagu s atributem `href`, ale React je s tímto obeznámen a neposílá další požadavky na server.

4 VÝSLEDKY ŘEŠENÍ, UŽIVATELSKÝ MANUÁL

Výstupem projektu je funkční webová aplikace, která je rozdělená na uživatelskou část řešenou Reactem a administrační část, která je zpracována ve Flasku. Webové stránky jsme se snažili udělat co nejvíce uživatelsky přívětivé, tak, aby návštěvník našel ihned to, co hledá – bez zbytečných reklam a vyskakovacích oken.

4.1 Cíle projektu

Splněné cíle:

- Vytvoření vizuálního návrhu ve Figmě,
- tvorba datového modelu,
- frontend webové aplikace,
- backend webové aplikace,
- automatické vytváření obrázků a jejich následné nahrávání na sociální sítě.

Rozpracované cíle:

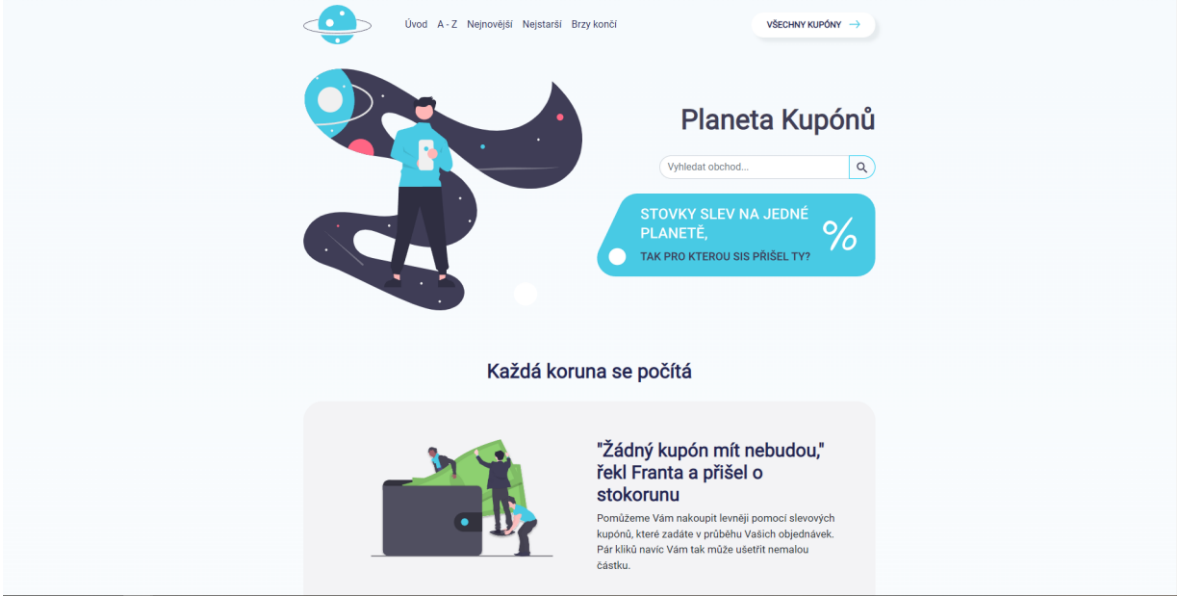
- Vytvoření skriptu, který by sbíral volně dostupné kódy a automaticky je ukládal do databáze,
- desktopová aplikace pro zobrazování kupónů.

4.2 Uživatelský manuál

4.2.1 Běžný uživatel

Po načtení stránky se uživateli zobrazí úvodní sekce s vyhledavačem obchodů. Pokud na stránku nepřichází uživatel s jasným cílem, může využít položek navigace. Tyto položky umožňují zobrazit všechny kupóny seřazeny dle výběru.

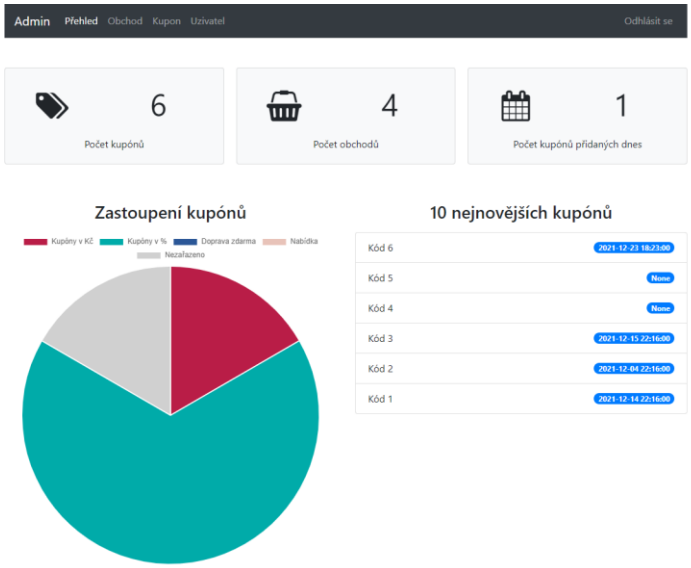
Níže na úvodní stránce se nachází informační banner a přehled 9 nejnovějších kupónů.



Obrázek 4.2.1: Zobrazení pro běžného uživatele

4.2.2 Uživatel s administrátorskými právy

Při načtení administrátorské stránky (/admin) se uživateli zobrazí přihlašovací formulář. Pokud uživatel správně zadá uživatelské jméno a heslo, je přesměrován na stránku s panelem ke správě dat na webu. První záložka obsahuje statistiky, pomocí dalších lze přidávat a editovat uživatele, obchody i jednotlivé kupóny.



Obrázek 4.2.2: Zobrazení pro administrátora

ZÁVĚR

Hlavním cílem projektu bylo vytvořit plně automatizovanou webovou aplikaci, která by shromažďovala slevové kupóny. Cíl byl až na pár maličkostí a nedokončených částí (např. automatizace získávání kupónů) dosažen, ovšem tyto nedostatky bychom v budoucnu rádi opravili a webovou aplikaci tak dotáhli až do úplného konce.

Výsledkem snažení je poloautomatizována webová aplikace, kde frontendová část je tvořena pomocí javascriptové knihovny React a frameworku Bootstrap. Backend projektu stojí na mikro frameworku Flask doplněn rozšířeními Flask-RESTful pro jednoduchou práci s API a Flask Admin, které posloužilo pro vytvoření administračního prostředí.

Do budoucna se máme v plánu nadále věnovat projektu, jak už nedokončeným částem, tak jeho případné publikaci veřejnosti.

Projekt jako celek mi přinesl mnoho nových zkušeností, seznámil jsem se s technologiemi, se kterými jsem neměl do této doby ještě čest a doufám, že tyto nově nabyté informace zúročím v dalších projektech.

Praktické řešení: https://github.com/PlanetaKuponu/web

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Bootstrap. *Build fast, responsive sites with Boostrap* [online], getbootstrap.com. [cit. 2021-12-30]. Dostupné z https://getbootstrap.com/.

[2] FARRELL, Doug. *Python REST APIs With Flask, Connexion, and SQLAlchemy* [online], realpython.com [cit. 2021-12-30]. Dostupné z https://realpython.com/flask-connexion-rest-api/.

[3] Exordium. *How to Setup Electron With React and TailwindCSS* [online], YouTube. 4. 9. 2021 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=ZsjgueodULk&ab_channel=Exordium.

[4] FlaskRESTful. *FlaskRESTful API* [online], flask-restful.readthedocs.io. [cit. 2021-12-30]. Dostupné z https://flask-restful.readthedocs.io/en/latest/.

[5] freeCodeCamp.org. *Full React Course 2020 - Learn Fundamentals, Hooks, Context API, React Router, Custom Hooks* [online], YouTube. 6. 10. 2020 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=4UZrsTqkcW4&ab_channel=freeCodeCamp.or.

[6] JavaScriptTutorial. *JavaScript Fetch API* [online], javascripttutorial.net. [cit. 2021-12-30]. Dostupné z https://www.javascripttutorial.net/javascript-fetch-api/.

[7] MDN Web Docs. *Using Fetch* [online], developer.mozila.org. 12. 10. 2021 [cit. 2021-12-30]. Dostupné z https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch.

[8] Pretty Printed. *Build a User Login System With Flask-Login, Flask-WTForms, Flask-Bootstrap, and Flask-SQLAlchemy* [online], YouTube. 2. 3. 2017 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=8aTnmsDMldY&ab_channel=PrettyPrinted.

[9] Pretty Printed. *Flask-Admin - An Example With an Existing Data Model* [online], YouTube. 8. 12. 2016 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=0cySORIhkCg&ab_channel=PrettyPrinted.

[10] Pretty Printed. *How to Integrate Flask-Admin and Flask-Login* [online], YouTube. 3. 4. 2018 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=NYWEf9bZhHQ&ab_channel=PrettyPrinted.

[11] ReactJS. *Components and Props* [online], reactjs.org. [cit. 2021-12-30]. Dostupné z https://reactjs.org/docs/components-and-props.html.

[12] Red Hat. *What is a REST API?* [online], redhat.com. 8. 5. 2020 [cit. 2021-12-30]. Dostupné z https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[13] Skolo Online. *Automate Instagram Posts with Python and Instagram Graph API* [online], YouTube. 20. 3. 2021 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=Q5kw7vGLqgs&ab_channel=SkoloOnline.

[14] The Net Ninja. *Full React Tutorial* [online], YouTube. 22. 1. 2021 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d&index=1&ab_channel=TheNetNinja.

[15] THE SHOW. *Docker Tutorial for Beginners Full Video - Learn Docker By Building A Simple Flask React Application* [online], YouTube. 3. 4. 2021 [cit. 2021-12-30]. Dostupné z https://www.youtube.com/watch?v=ByUWienlDDA&ab_channel=THESHOW.