SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE
dokumentace

Planeta kupónů

Lukáš Rychlý

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

Tato práce se zaměřuje na tvorbu webové aplikace, která má za úkol zprostředkovat uživatelům slevové kupóny z různých e-shopů.

V první části se zaměřuje na teoretický popis a definici pojmů, které jsou nutné k pochopení třetí části. Druhá část popisuje využité technologie a zdůvodnění, proč byly použity. Zmíněná třetí část se věnuje popisům jednotlivých částí aplikace, zmiňuje způsoby řešení a zachycuje útržky kódů a obrázky. Poslední část nabízí uživatelský manuál a výčet splněných a rozpracovaných cílů. Závěr je zaměřen na shrnutí výsledků, subjektivní zhodnocení práce a zmiňuje možná budoucí vylepšení aplikace.

Na celém projektu se podíleli tři lidé (Lukáš Rychlý, Jan Čech, Matěj Čech), v této dokumentaci jsou tak některé pasáže psány v množném čísle.

OBSAH

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 TEORETICKÁ A METODICKÁ VÝCHODISKA

1.1 Databáze

Pod pojmem databáze si můžeme představit systém souborů a určitý softwarový systém, který nám umožňuje pracovat s daty a ukládat je.

1.1.1 Databázový model

Definuje schéma databáze, určuje, jakým způsobem budou data organizována a strukturována. Existuje mnoho různých druhů databázových modelů, mezi nejstarší a dnes již nepoužívané řadíme například modely hierarchické a síťové. Jelikož tyto modely nebyly zcela optimální, byly postupem času nahrazeny zejména relačními a objektovými databázovými modely.

1.1.2 Vazby mezi tabulkami relačních databází

Vazba 1:1

Vazba, při které jedna položka v první tabulce odpovídá právě jedné položce v tabulce druhé.

Vazba 1:N

Spojení tabulek, v kterém jedna položka první tabulky odpovídá n-položkám v druhé tabulce.

Vazba M:N

Vztah mezi tabulkami, při kterém m-položek v první tabulce odpovídá n-položkám v druhé tabulce.

1.1.3 Primární a cizí klíče

Primární klíče se používají jako unikátní identifikátory záznamu tabulky. Jedná se převážně o číselné řady. Cizí klíče používáme k vyjádření vztahů mezi tabulkami.

1.1.4 Základní datové typy

Integer – celé číslo. **Varchar** – kratší text. **Text** – delší text. **Boolean** – true/false.

1.2 Frontend

Jedná se o prezentační vrstvu webových stránek. Zjednodušeně řečeno je to část webových stránek viditelná pro všechny uživatele.

1.2.1 Komponenty

Aby aplikace nebyla pouze nepřehledný celek, je žádoucí rozdělit ji na dílčí části, které budou tvořeny tzv. komponenty. Tyto komponenty můžeme v Reactu vytvořit pomocí tříd nebo funkcí. Často se jedná o různá tlačítka, nadpisy, bloky atd. Výhodou těchto komponentů je jejich možnost opakovaného použití.

1.2.2 Bootstrap

Bootstrap je nejpoužívanější frontend framework pro vývoj webových aplikací. Jedná se o předem připravenou sadu stylů, kterou následně můžeme aplikovat na HTML kód. Jeho hlavní výhodou je jednoduché zajištění responzivity.

1.3 Backend

Backend tvoří administrační vrstvu webových stránek. Jedná se o část webových stránek, ke které nemá běžný uživatel přístup, slouží zejména ke správě webové aplikace. Abychom se k této části dostali, je nutné provést autorizaci (přihlášení).

1.3.1 Framework

Framework je určitá struktura (sada vývojářských nástrojů), která programátorům usnadňuje vývoj aplikací. Většinou obsahuje knihovny a další software, které může programátor využít a značně si tak zjednodušit svou práci.

1.3.2 API

Pojmem API (Application Programming Interface) se rozumí rozhraní pro programování aplikací. Zjednodušeně řečeno API funguje jako prostředník mezi dvěma platformami – zajišťuje mezi nimi vzájemnou komunikaci.

1.3.3 Administrační prostředí

Slouží administrátorům k jednoduché správě webových aplikací. Často je tvořeno lištou panelů, které umožňují práci s databází, uživateli i konkrétními daty. Může obsahovat různé statistiky, které se týkají webových stránek (návštěvnost, prokliky, počet zakoupených výrobků atd.).

1.3.4 Zabezpečení

Aby se běžní uživatelé nedostali do administrační části webových stránek (backendu), je nutné provést určité kroky, které jim v tom zabrání. Pokud uživatel zažádá o přístup ke správě webové aplikace, je nutné provést autentizaci (proces ověření identity subjektu) – nejčastěji pomocí přihlašovacího formuláře. V případě, že je autentizace úspěšná, následuje autorizace, která již danému uživateli umožní přístup do administračního prostředí.

1.4 Docker

Virtualizační technologie, která pracuje s kontejnery.

1.4.1 Kontejnery

Kontejner obsahuje aplikaci a soubory, které daná aplikace nezbytně vyžaduje. Neobsahuje však operační systém, který je u běžné virtualizace přítomen. Díky tomu jsou výrazně sníženy režijní náklady. Takto vzniklé kontejnery sdílejí jeden běžící operační systém. Jejich výhodou je rychlé spuštění a izolace od daného prostředí.

1.5 Automatizace obrázků

V našem podání se jedná o automatické vygenerování obrázků z dostupných kupónů a jejich následné zaslání na sociální sítě bez zásahu administrátora.

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).

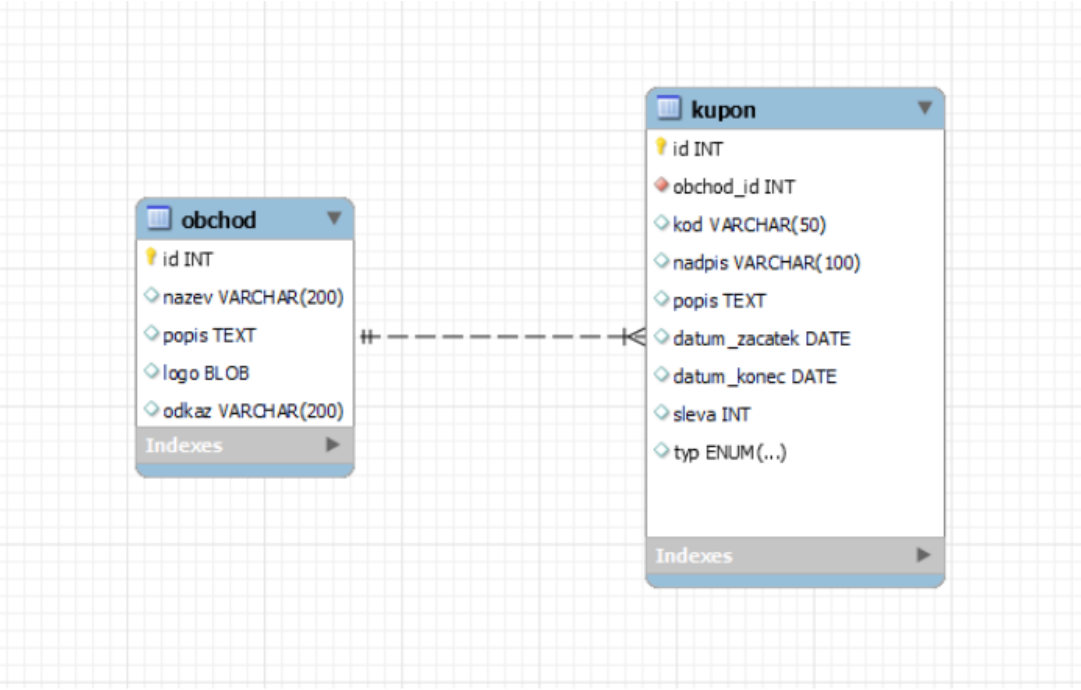
Obrázek 2.6: Docker

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

3.1 Databáze

3.1.1 Datový model

Výchozí tabulkou námi vytvořeného datového modelu je *Kupon*. Této tabulce je předáván cizí klíč z tabulky *Obchod*. Jedná se o vztah 1:N, tedy jeden obchod může být přiřazen k více kupónům.



Obrázek 3.1.1: Schéma datového modelu

Další důležitou tabulkou je *Uzivatel*. Tato tabulka, jak je již patrné z názvu, pracuje s uživateli. Primárním klíčem je *id* s automatickou inkrementací. Dalšími atributy jsou *jmeno* a *heslo*, které jsou nezbytně nutné pro následnou autentizaci uživatelů.

```
class Uzivatel(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    jmeno = db.Column(db.String(15), nullable=False, unique=True)
    heslo = db.Column(db.String(200), nullable=False)
```

Ukázka 3.1.1: Tabulka *Uzivatel*

3.2 Frontend

3.2.1 Komponenty

Veškeré komponenty jsou umístěny v následující adresářové struktuře:

web/frontend/src/components/.

K tvorbě jednotlivých komponentů jsme využili knihovnu React Bootstrap, JSX syntaxi a vlastní CSS styly.

Pro popis komponentu jsem si vybral *Kuponek.jsx*. Při tvorbě tohoto komponentu jsem vycházel z grafického návrhu na Figmě. Jako základ jsem si zvolil komponent *Card* z knihovny React Bootstrap, kterou jsem na začátku patřičně importoval. Aby docházelo ke střídání barev kupónů, bylo nutné skrze ternární operátor přidávat konkrétní CSS třídu na základě zadaného parametru (*kuponBarva*). Dalšími předávanými parametry tohoto komponentu jsou: *nazevObchodu*, který pomocí API přiřazuje název obchodu, výše slevy je vyjádřena parametrem *sleva*, parametr *kod* v komponentu slouží k vypsání kódu, který se zadává do e-shopů, a poslední parametr *text* informuje uživatele o podmínkách uplatnění slevového kupónu.

```
return(
<Card className={"kupon " + (kuponBarva === "modra" ? 'modry-kupon' : 'bily-kupon')}>
    <Card.Body>
      <Card.Title className="obchod">{nazevObchodu.nazev}</Card.Title>
      <Card.Subtitle className="mb-2 sleva">{sleva}</Card.Subtitle>
      <Card.Text className="kod">
        KÓD: {kod}
      </Card.Text>
      <Card.Text>
        {text}
      </Card.Text>
    </Card.Body>
  </Card>
)
```

Ukázka 3.2.1: Část komponentu *Kuponek.jsx*

Pokud jsou všechny parametry předány správně a nic není opomenuto, bude výstupem komponent následujícího vzhledu:

Obrázek 3.2.1: Výstup komponentu *Kuponek.jsx*

3.3 Backend

3.3.1 API

Při vytváření API jsme se rozhodli použít knihovnu Flask-RESTful společně s knihovnou Flask-Marhsmallow. Samotná tvorba API je s těmito knihovnami poměrně jednoduchá. Nejprve je nutné vytvořit třídu, která dědí parametry třídy *Resource*, která je naimportovaná z knihovny Flask-RESTful. V takto vzniklé třídě se nachází funkce, která po selekci dat pomocí SQL dotazů a schémat (vytvořených pomocí Flask-Marhsmallow) vrací požadovaná data v JSON formátu. Pomocí metody *add_resource* následně spojíme třídu s požadovaným endpointem.

```
class Kupony_nejnovejsi(Resource):
    def get(self):
        kupony = Kupon.query.order_by(desc(Kupon.id)).all()
        kupon_schema = KuponSchema(many=True)
        return kupon_schema.dump(kupony)
```

Ukázka 3.3.1: Část kódu z *api.py*, API pro výpis nejnovějších kupónů

3.3.2 Zabezpečení

Abychom zajistili bezpečnost a zabránili nechtěnému vniknutí do administrační části webové aplikace, rozhodli jsme se využít knihovnu Flask-Login společně s knihovou Flask-Admin. Nejprve bylo nutné vytvořit strukturu uživatele v *models.py*, která je tvořena atributy *id*, *jmeno* a *heslo*. Aby bylo přihlašování uživatelsky přívětivé, vytvořili jsme ve složce *templates* dva HTML soubory – jeden pro přihlášení, druhý pro přidání nového uživatele do databáze. Pomocí knihovny Flask-WTF jsme rovněž připravili dva formuláře k zmíněným HTML souborům.

Funkčnost zabezpečení je následně zařízena v souboru *routes.py*. Pokud se chceme přihlásit, je nejprve ověřena existence uživatele v databázi, poté dojde ke kontrole zahashovaného hesla. V případě, že se zadané údaje shodují s daty v databázi, je uživatel přesměrován do administračního prostředí. V opačném případě je zobrazena chybová hláška a je vyžadováno opakované zadání údajů. Jakmile je uživatel přihlášen, může libovolně nakládat s daty pomocí zmíněného administračního prostředí, rovněž může vytvářet nové administrátorské účty.

```
@app.route('/prihlaseni', methods=['GET', 'POST'])
def prihlaseni():

    if current_user.is_authenticated:
        return redirect('/admin')

    form = LoginForm()

    if form.validate_on_submit():
        user = Uzivatel.query.filter_by(jmeno=form.jmeno.data).first()
        if user:
            if check_password_hash(user.heslo, form.heslo.data):
                login_user(user, remember=form.zapamatovat.data)
                return redirect('/admin')

        return '<h1>Špatně zadané uživatelské jméno nebo heslo</h1>'

    return render_template('prihlaseni.html', form=form)
```

Ukázka 3.3.2: Část kódu z *routes.py*, přihlašování do administračního prostředí

3.4 Automatizace obrázků

3.4.1 Requesty, Instagram Graph API

Aby bylo možné zasílat obrázky automaticky na Instagram, museli jsme si nejprve založit vývojářský účet na stránce https://developers.facebook.com/. V rámci tohoto účtu jsme vytvořili aplikaci, která generuje ověřovací tokeny. Pomocí těchto tokenů je následně možné zasílat smysluplné požadavky na API Facebooku a Instagramu. Pokud je požadavek sepsán správně, můžeme skrze něj automaticky přidávat obsah na zmíněné sociální sítě.

V Pythonu jsme vyřešili zasílání požadavků knihovnou Requests a patřičnou funkcí. Jelikož je zasílání požadavků omezeno na 25 requestů za den, je funkce řízena cyklem. Tato funkce se následně bude spouštět každý den pomocí Cronu.

```
def postInstagram():
    for i in range(config.pocet_obrazku):
        obrazek = oldest_file()
        if obrazek is not None:
            post_url =
'https://graph.facebook.com/v12.0/{}/media'.format(config.ig_user_id)
            payload= {
            'image_url':
'https://res.cloudinary.com/demo/image/upload/v1312461204/sample.jpg',
            'caption': 'slevový kupón',
            'access_token': config.user_access_token
            }

            r = requests.post(post_url, data=payload)

            result = json.loads(r.text)

            if 'id' in result:
                creation_id = result['id']
                second_url =
'https://graph.facebook.com/v12.0/{}/media_publish'.format(config.ig_user_id)
                second_payload = {
                    'creation_id': creation_id,
                    'access_token': config.user_access_token
                }
```

```
                r = requests.post(second_url, data=second_payload)

            os.remove(obrazek)

        else:
            break
```

Ukázka 3.4.1: Část kódu z *instagram.py*, automatické přidávání obrázků na Instagram

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:

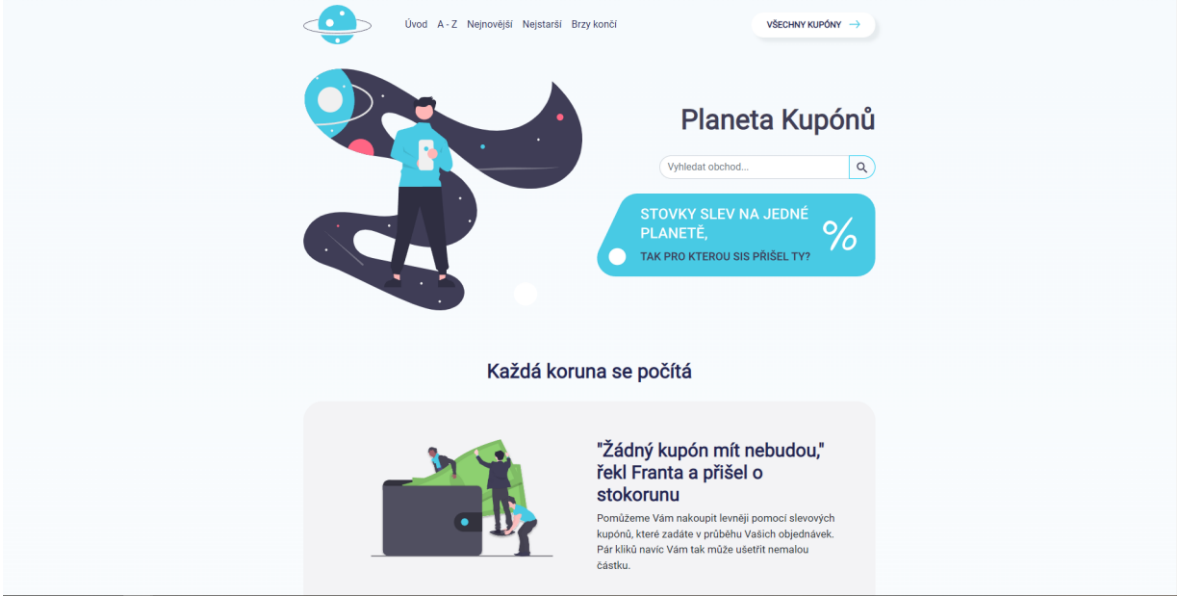
- Vytvoření skriptu, který by sbíral volně dostupné kódy a automaticky je ukládal do databáze,
- desktopová aplikace pro zobrazování kupónů.

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.

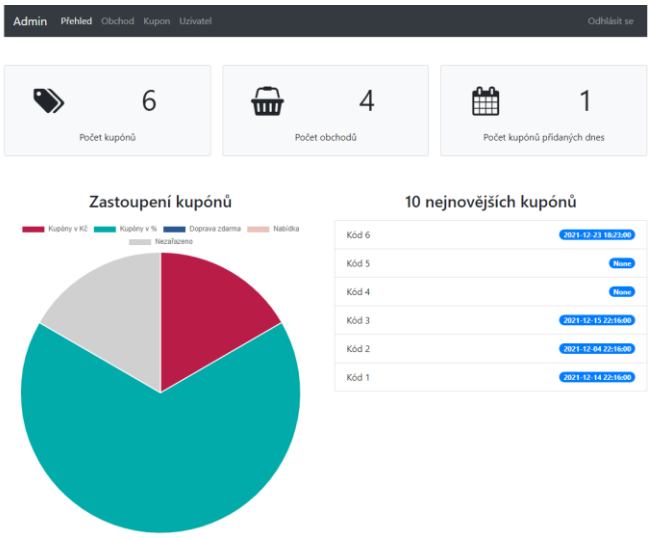
Níže na úvodní stránce se nachází informační banner a přehled 9 nejnovějších kupónů.



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

Myslím si, že hlavní cíl naší práce byl splněn. Vytvořili jsme funkční webovou aplikaci, která lidem umožňuje vyhledávat slevové kupóny.

Věřím, že projekt nalezne uplatnění v praxi, a to hned z několika důvodů. Je přehledný, uživatel ihned nalezne to, co právě potřebuje. Na rozdíl od konkurence neobsahuje všudypřítomné reklamy. A v neposlední řadě je žádaný, protože Češi zkrátka slevy milují.

Vylepšení se nabízí vícero. Mé přání bylo, aby byly webové stránky plně automatizované. To se nám však bohužel nepovedlo – nyní lze kupóny přidávat pouze ručně. Mezi další možná vylepšení bych zařadil například tvorbu mobilní aplikace.

Celý projekt vnímám jako příležitost, která mi přinesla mnoho nových poznatků a zkušeností z IT světa.

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.