

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Planeta kupónů

Matěj 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

Práce popisuje zčásti zautomatizovanou full stack webovou aplikaci, která se zaměřuje na zprostředkování a následné zpracování slevových kupónů z různých internetových obchodů.

Backend projektu stojí na mikro frameworku Flask 2.0.1, který kupříkladu umožňuje definovat datový model, operovat s daty a přidávat endpointy API.

Funkčnost REST API je zajištěna rozšířením Flask-RESTful 0.3.9. Tato technologie zajišťuje předávání uložených dat, jako jsou například záznamy všech kupónů, ve formátu JSON.

Frontend – část webové aplikace, která je uživateli dostupná použitím webového prohlížeče – je zpracován JavaScriptovou knihovnou React.js 17.0.2, jelikož je vhodná pro práci s rychle se měnícími daty. Zpracovávaná data jsou získávána zasíláním dotazů na příslušné endpointy API.

Na celém projektu se podíleli tři lidé (Matěj Čech, Jan Čech, Lukáš Rychlý).

OBSAH

ÚVOD.....	6
1 BACKEND.....	7
1.1 FRAMEWORK.....	7
1.2 API	7
1.2.1 REST API.....	7
1.2.2 Endpoint	8
1.2.3 Schéma	8
1.3 TEMPLATES	8
1.4 DEKORÁTOR ROUTE	9
1.5 ADMINISTRAČNÍ PROSTŘEDÍ	9
1.6 ZABEZPEČENÍ	9
1.7 AUTOMATIZACE OBRÁZKŮ	10
1.7.1 Generování obrázků	10
2 VYUŽITÉ TECHNOLOGIE	11
2.1 FIGMA	11
2.2 REPLIT.....	11
2.3 CLOCKIFY	12
2.4 REACT.JS	12
2.5 FLASK	13
2.6 POSTGRESQL.....	13
2.7 DOCKER & DOCKER COMPOSE.....	13
2.7.1 Kontejnery	14
3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY.....	15
3.1 DATABÁZE	15
3.1.1 Datový model	15
3.1.2 Importování vzorových dat	16
3.2 REST API.....	16
3.2.1 Vytvoření schéma.....	17
3.2.2 Vytvoření metody pro získávání dat	17
3.2.3 Přidání zdroje	18
3.2.4 Vyhledávání kupónů pomocí API.....	18
3.3 ADMINISTRAČNÍ PROSTŘEDÍ	18
3.3.1 Inicializace tabulky	18
3.3.2 Stránka se statistikami.....	19
3.3.3 Zabezpečení.....	20
3.3.4 Přidání administrátorského uživatele	22

3.4	AUTOMATIZACE OBRÁZKŮ	22
3.4.1	Generování obrázků pro sociální sítě	22
3.4.2	Generování obrázků z administračního prostředí.....	23
4	VÝSLEDKY ŘEŠENÍ, UŽIVATELSKÝ MANUÁL	24
4.1	CÍLE PROJEKTU	24
4.2	UŽIVATELSKÝ MANUÁL	24
4.2.1	Běžný uživatel	24
4.2.2	Uživatel s administrátorskými právy	25
	ZÁVĚR	26
	SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ	27

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 našli 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.

Zaměření jednotlivých částí dokumentace

První část dokumentace popisuje teoretickou část projektu a definuje různé pojmy, které jsou stěžejní pro pochopení realizace projektu. Druhá část se zaměřuje na využití technologie, odůvodňuje jejich výběr a užití. Část třetí obsahuje konkrétní popis jednotlivých částí webové aplikace a ukazuje jejich řešení v podobě části příslušného kódu. Čtvrtá část poukazuje na cíle projektu a zprostředkovává uživatelský manuál. Závěr této práce hodnotí cíle projektu a jejich zpracování, uvádí využití v praxi a zmiňuje možná budoucí vylepšení.

1 BACKEND

Backend je část webové aplikace, která operuje se samostatnými daty a zprostředkovává je frontendu. K této části nemá běžný návštěvník stránek přístup.

1.1 Framework

Cílem frameworku je usnadnění vývoje různých aplikací tak, že poskytuje konkrétní funkce a řeší mnohé problémy v oblasti, pro kterou byl vytvořen. Programátorovi tak umožňuje se zaměřit na specifické funkce a další záležitosti vyvíjené aplikace.

Příkladem námi využitého frameworku je právě Flask. Ten umožňuje vyvíjet aplikace bez nutnosti se starat o nízko úrovněvé detaily. Konkurencí Flasku je kupříkladu Django, které však již zpočátku obsahuje nástroje, které by v našem případě byly zbytečné.

Hlavním důvodem, proč jsme zvolili zrovna Flask, je jeho malé jádro a snadná škálovatelnost. Flask v základu neobsahuje některé funkce oproti Django, jako je například administrátorské prostředí nebo možnost multijazyčnosti. Tyto rozšíření a mnoho dalších však lze v případě potřeby dodatečně do struktury Flasku implementovat.

1.2 API

API je sada definic a protokolů pro vytváření a integraci aplikačního softwaru, slouží ke komunikaci se systémem za účelem získání informací či provedení nějaké funkce. API pomáhá systému sdělit požadavek tak, aby mu porozuměl a následně jej mohl splnit. Rozhraní API si tak můžeme představit jako prostředníka mezi uživatelem a službou.

1.2.1 REST API

REST API je architektonický styl navržený pro systémy, jako je například WWW. Když klient provede dotaz skrze RESTful API, vrátí se mu reprezentace stavu. Tyto informace jsou předávány v jednom z několika formátů prostřednictvím HTTP. Příkladem nejužívanějšího formátu je JSON, jelikož je dobře čitelný jak pro lidi, tak pro stroje.



Obrázek 1: Znázornění funkce REST API

1.2.2 Endpoint

Jednoduše řečeno, endpoint je jeden z koncových bodů komunikačního kanálu a je identifikován pomocí URL.

Každý endpoint je místo, které umožňuje rozhraní API přistupovat k požadovaným zdrojům. Když API provede dotaz, obdrží odpověď od serveru. Tato odpověď je poté zpracována a vrácena na místo, které se nazývá endpoint.

1.2.3 Schéma

Schéma jsou metadata, která určují, jak jsou data strukturovaná. Vytvořené schéma tak umožňuje jednoduší použití výsledného API, jelikož výstupem jsou zformátovaná data do požadovaného výstupu, jako je JSON nebo XML.

Jednotlivé endpointy API mohou být implementovány i bez použití schéma. Tímto přístupem by se však zkomplikovalo čtení dat vrácených z API, a to nejen člověku, ale i stroji.

1.3 Templates

Template neboli šablona je soubor, který obsahuje statická data a také zástupné symboly pro dynamická data. Po dosazení požadovaných dat do šablony se vytvoří konečný dokument. Flask využívá k vykreslování šablon šablonovací jazyk jménem Jinja.

1.4 Dekorátor route

Tento dekorátor slouží k nastavení požadované funkce, která se provede, pokud na URL adresu definovanou v dekorátoru `@route()` přijde nějaký požadavek.

URL adresa může také obsahovat sekce s proměnnými. Definovaná funkce následně získává tyto proměnné jako argumenty a může s nimi dále pracovat. Volitelně se může použít i konvertor k určení typu proměnné.

Tento dekorátor je důležitý, pokud potřebujeme vytvořit dostupnou stránku na straně backendu. Příkladem může být administrační prostředí nebo stránka obsahující přihlašovací formulář.

1.5 Administrační prostředí

Jedná se o prostředí, ve kterém si může oprávněný uživatel zobrazit data, upravit je nebo je i smazat. Pro přístup do tohoto prostředí je většinou nutná autentizace předáním validních přístupových údajů. Pokud autentizace proběhne v pořádku, následuje autorizace uživatele neboli umožnění přístupu a přesměrování do administračního prostředí.

1.6 Zabezpečení

K tomu, aby se neautorizovaný uživatel nedostal k informacím, ke kterým nechceme, aby měl přístup, je nutné tato data zabezpečit. To platí i při přenosu dat mezi klientem a serverem.

K omezení přístupu neoprávněným uživatelům na stránky, jako je administrační prostředí, můžeme využít přihlašovací formulář. Pokud se uživatelem zadané hodnoty shodují s požadovanými údaji, je mu umožněno pokračovat dále.

Zabezpečenou komunikaci mezi účastníky je možné zařídit použitím protokolu SSL. Tento protokol poskytuje zabezpečení komunikace užitím šifrování a autentizací komunikujících stran.

1.7 Automatizace obrázků

Automatizace v jakémkoliv ohledu odstraňuje psychickou námahu a snižuje potřebu přítomnosti člověka. Tak je tomu i v případě, pokud chceme dosáhnout automatického generování obrázků a následného odesílání těchto obrázků na sociální síť.

1.7.1 Generování obrázků

Jak již název napovídá, při tomto procesu dochází k vytváření obrázků, které obsahují informace určité problematiky. Vyhotovené obrázky jsou následně uloženy pro další zpracování.

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: 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 3: Replit

2.3 Clockify

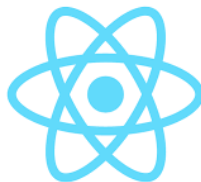
Clockify je online aplikace sloužící k sledování času a vytváření časového rozvrhu. Použili jsme ji jako záznamník námi vykonané práce. Na základě statistik stráveného času automaticky převádí zaznamenané hodnoty do přehledných grafů.



Obrázek 4: 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 5: 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 6: 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 7: 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 8: Docker

2.7.1 Kontejnery

Kontejner obsahuje pouze zabalený kód se všemi jeho závislostmi a dalšími specifickými soubory, avšak neobsahuje virtualizovaný operační systém. Díky tomuto aplikace běží rychle, má menší velikost a náklady na provoz jsou nižší.

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

Cílem projektu bylo vytvořit full stack webovou aplikaci, která bude zprostředkovávat slevové kupóny. Aby se tohoto cíle dosáhlo, bylo nutné rozložit celý proces tvorby do několika etap, to platí i pro backend.

Hlavní etapy tvorby backendu:

1. Definování datového modelu,
2. vytvoření REST API, které by zprostředkovávalo potřebná uložená data,
3. vytvoření administračního prostředí pro jednoduchou správu dat,
4. zabezpečení administračního panelu,
5. zautomatizování generování obrázků.

3.1 Databáze

Data jsou do databáze ukládána pomocí rozšíření Flask-SQLAlchemy 2.5.1, které umožňuje spolupracovat s databázovým systémem PostgreSQL. Propojení je provedeno přiřazením odkazu databáze do proměnné `app.config['SQLALCHEMY_DATABASE_URI']`.

3.1.1 Datový model

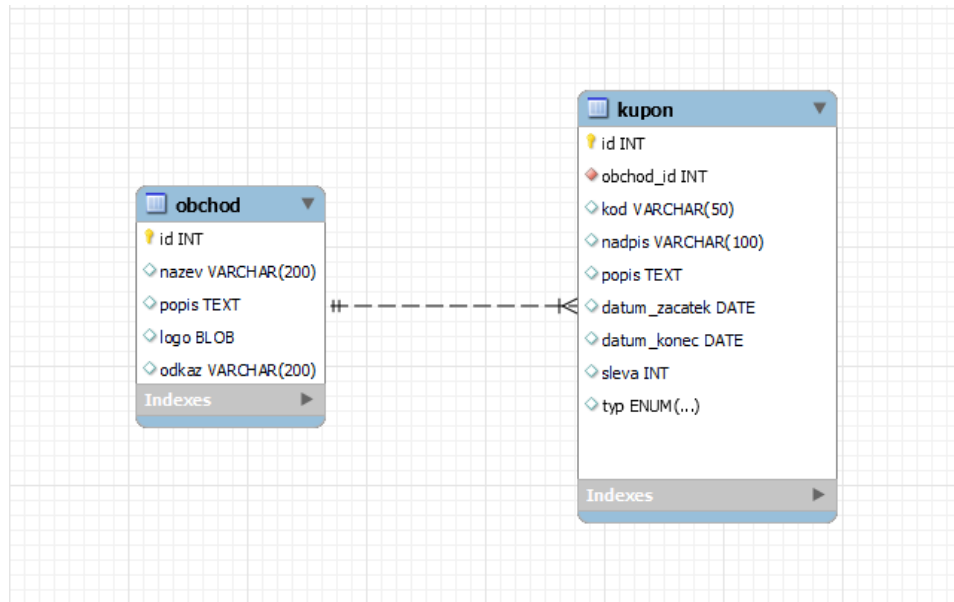
Datový model obsahuje tři tabulky – *obchod*, *kupon* a *uzivatel*.

Tabulka *obchod* obsahuje informace o jednotlivých obchodech, jako je například název, popis, odkaz a jednotlivé kupóny ve vztahu 1:N. To znamená, že jednotlivé obchody mohou mít několik vlastních kupónů.

V tabulce *kupon* jsou uloženy kupóny kupříkladu s atributy cizí klíč obchodu, kód, typ slevy a datum začátku. Položka *typ* je uložena v datovém typu zvaném enum neboli výčtový typ, tudíž se vybírá jeden typ z těchto předem definovaných hodnot: *doprava*, *procenta*, *koruny*, *nabídka*.

Schéma *uzivatel* není s dalšími tabulkami nijak provázané. Slouží k ukládání přístupových údajů administrátorských účtu v podobě jména a hesla v zahashované podobě.

K vygenerování tohoto hashe je použita funkce `generate_password_hash()` z knihovny Werkzeug 2.0.2, do které je společně s původním heslem předán typ hashovací funkce SHA256.



Obrázek 9: Znáznornění tabulek obchod a kupon

3.1.2 Importování vzorových dat

Aby se dala zobrazit nějaká data ve frontendu bez ručního vložení, nabízí se možnost importování uložených vzorových dat příkazem. Pro úspěšné importování je důležité, aby byl příkaz proveden z místa, ve kterém se nachází soubor *docker-compose.yml*.

Pro import vzorových dat je možné použít tento příkaz:

```
$ cat db/dump/sample_data.gz | gunzip | docker-compose exec -T db psql -U data-base -d planeta_database
```

3.2 REST API

Flask-RESTful 0.3.9 je rozšíření, které umožňuje rychlé vytváření REST API.

3.2.1 Vytvoření schéma

Pomocí endpointů chodí dotazy a REST API vrací požadovaná data. Aby však posílaná data byla správně ve formátu JSON, je nutné je upravit podle předem definovaného schématu. K tomuto účelu slouží knihovna Flask-Marshmallow 0.14.0, která zároveň umožňuje vybírat zobrazovaná data.

V schématu musí být v proměnné `model` definovaný datový model, ze kterého budou data vybírána. V proměnné `fields` se mohou nastavit požadovaná data zvoleného modelu namísto všech. My jsme si u schéma `KuponSchema`, zvolili pouze některá data, jelikož například s datumem začátku platnosti kupónu nepotřebujeme ve frontendu pracovat.

Ukázka kódu – schéma pro kupóny:

```
class KuponSchema(ma.Schema):
    class Meta:
        model = Kupon
        fields = ("id", "obchod_id", "kod", "nadpis", "popis", "sleva", "datum_konec")
        sqla_session = db.session
```

3.2.2 Vytvoření metody pro získávání dat

Základním stavebním blokem je třída `Resource` z Flask-RESTful, která umožňuje jednoduché vytvoření více metod jejich definováním u své vlastní třídy. My jsme však pouze využili jednu metodu a tou je `GET`.

Pro výběr potřebných dat je využita metoda `query`, následně jsou tyto data filtrovány funkcí `filter()` a potřebným filtrujícím výrazem.

Ukázka kódu – metoda pro získávání dat hledaných kupónů:

```
class Kupony_search(Resource):
    def get(self, search):
        kupony = Kupon.query.join(Obchod, Kupon.obchod_id==Obchod.id).filter(Obchod.nazev.ilike(f"%{search}%")).all()
        kupon_schema = KuponSchema(many=True)
        return kupon_schema.dump(kupony)
```

3.2.3 Přidání zdroje

Aby bylo vůbec možné k datům pomocí API přistupovat, je nutné pomocí metody `add_resource()` přidat vytvořenou třídu jako zdroj.

Ukázka kódu – přidání funkce hledání kuponů jako zdroj:

```
api.add_resource(Kupony_search, '/api/vyhledat-kupony/<string:search>')
```

3.2.4 Vyhledávání kuponů pomocí API

Pro vyhledávání kuponů podle názvu obchodu stačí přejít na podstránku webové aplikace `/api/vyhledat-kupony/<vyhledávaný pojem>`. Vyhledávaný pojem nemusí být celý, stačí, aby se pojem shodoval s částí kteréhokoliv názvu obchodu.

3.3 Administrační prostředí

Rozšíření Flask-Admin 1.5.8 dokáže vytvořit jednoduché, avšak nezabezpečené administrační prostředí. Toto prostředí zpočátku umožňuje zobrazovat jednotlivé záznamy inicializovaných tabulek, měnit je nebo je dokonce i smazat. Prostor se ovšem dá rozšířit vlastním obsahem, a to i dynamickým.

Pro zobrazení administračního prostředí, stačí ve webovém prohlížeči přejít na podstránku webové aplikace `/admin` a vyplnit správné přístupové údaje. Při použití vzorových dat, se lze přihlásit jménem *uzivatel* a heslem *heslo1234*.

3.3.1 Inicializace tabulky

Při tvorbě modelového pohledu se vybírají položky z datového modelu, které bude možné v administračním prostředí definovat či upravovat. Tyto údaje se nastavují přiřazením seznamu obsahujícího požadované položky do proměnné `form_columns`. Celá tabulka se následně inicializuje pomocí funkce `add_view()`.

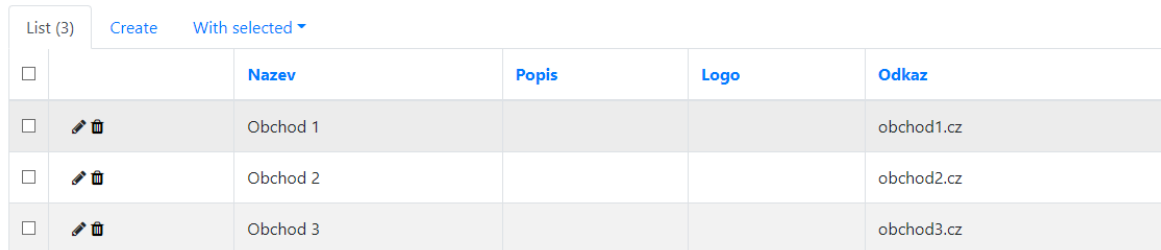
Ukázka kódu – vytvoření vlastního modelového pohledu pro obchody:







```
class ObchodView(ModelView):  
    form_columns = ['nazev', 'popis', 'odkaz']
```

Ukázka kódu – inicializace modelového pohledu pro obchody:

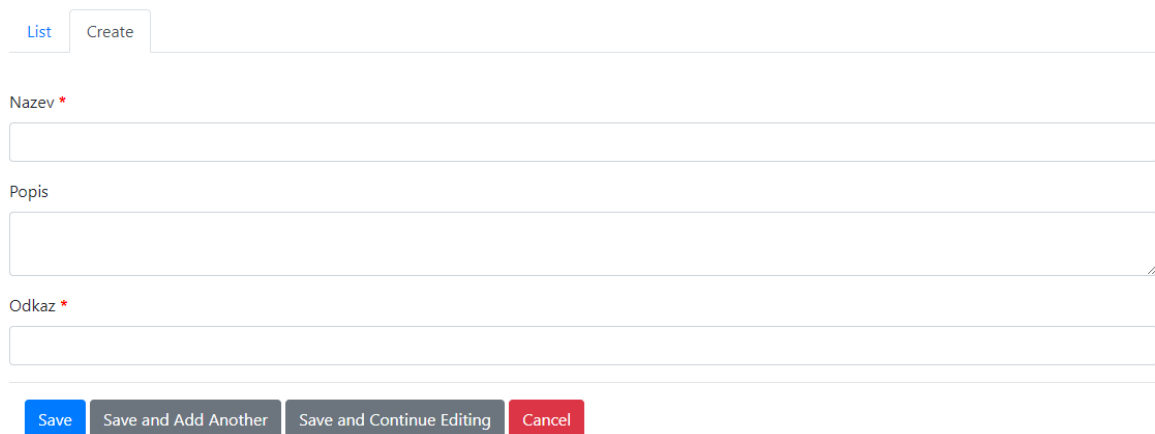
```
admin.add_view(ObchodView(Obchod, db.session))
```

Po inicializaci tabulky se vytvoří v administračním prostředí přehled již uložených záznamů. Tyto záznamy následně lze upravovat, mazat, nebo je také možné vytvořit úplně nový záznam.



List (3)	Create	With selected ▾			
<input type="checkbox"/>		Nazev	Popis	Logo	Odkaz
<input type="checkbox"/>	 	Obchod 1			obchod1.cz
<input type="checkbox"/>	 	Obchod 2			obchod2.cz
<input type="checkbox"/>	 	Obchod 3			obchod3.cz

Obrázek 10: Přehled uložených záznamů v tabulce Obchod



List Create

Nazev *

Popis

Odkaz *

Save Save and Add Another Save and Continue Editing Cancel

Obrázek 11: Formulář pro vytvoření nového záznamu v tabulce Obchod

3.3.2 Stránka se statistikami

Statistická data jsou zajištěna hledáním požadovaných dat v databázi, součtem jejich četnosti a následným předáváním do šablony, kde jsou vykreslována. K vyobrazení těchto dat je však nutné rozšířit administrační prostředí.

Aby bylo možné rozšířit hlavní stránku administračního panelu o vlastní obsah, je nutné:

1. Vytvořit vlastní pohled a definovat požadovaná data,
2. nastavit nově vytvořený pohled jako hlavní stránku administračního prostředí,
3. vytvořit šablonu, která zpracovává data definovaná v pohledu.

Ukázka kódu – vytvoření vlastního pohledu pro statistický panel:

```
class MyAdminIndexView(AdminIndexView):
    @expose('/')
    def index(self):
        today = datetime.today()
        return self.render('admin/index.html',
            pocet_kuponu=Kupon.query.count(),
            pocet_obchodu=Obchod.query.count(),
            pocet_kuponu_kc=Kupon.query.filter(Kupon.typ == "koruny").count(),
            pocet_kuponu_procento=Kupon.query.filter(Kupon.typ == "procenta").count(),
            pocet_kuponu_doprava=Kupon.query.filter(Kupon.typ == "doprava").count(),
            pocet_kuponu_nabidka=Kupon.query.filter(Kupon.typ == "nabidka").count(),
            pocet_kuponu_nezarazeno=Kupon.query.filter(Kupon.typ == None).count(),
            pocet_kuponu_dnes=Kupon.query.filter(
                extract('month', Kupon.datum_zacatek) == today.month,
                extract('year', Kupon.datum_zacatek) == today.year,
                extract('day', Kupon.datum_zacatek) == today.day).count(),
            nejnovější_kupony=Kupon.query.order_by(desc(Kupon.id)).limit(10))
```

Ukázka kódu – rozšíření hlavní stránky administračního prostředí vlastním pohledem:

```
admin = Admin(app, index_view=MyAdminIndexView(name='Přehled'),
    template_mode='bootstrap4')
```

Ukázka kódu – část šablony statistického panelu – počet kupónů:

```
<div class="card bg-light">
    <div class="card-body text-center">
        <div class="row">
            <div class="col-5"><i class="fa fa-tags pocet-ikona mt-3"></i></div>
            <div class="col-7 pocet-cislo">{{ pocet_kuponu }}</div>
        </div>
        <p class="card-text mt-3">Počet kupónů</p>
    </div>
</div>
```

3.3.3 Zabezpečení

Zabezpečení administračního panelu je dosaženo rozšířením Flask-Login 0.5.0. Toto rozšíření zajišťuje přihlášení, odhlášení a správu relací.

Přihlašovací formulář společně s formulářem pro vytvoření nového uživatele je vytvořen pomocí knihovny Flask-WTF 1.0.0.

Aby se mohl k administrátorskému panelu (/admin) dostat pouze přihlášený uživatel, je nutné přidat funkci `is_accessible()` do vlastního pohledu. Pro přesměrování nepřihlášeného uživatele na stránku pro přihlášení, se přidá funkce `inaccessible_callback()`.

Ukázka kódu – zabezpečení administrátorského panelu:

```
class MyAdminIndexView(AdminIndexView):
    @expose('/')
    def is_accessible(self):
        return current_user.is_authenticated

    def inaccessible_callback(self, name, **kwargs):
        return redirect('/prihlaseni')
```

Jelikož stránka pro vytvoření nového uživatele a odhlášení není přímo součástí administrátorského panelu, je potřeba je zabezpečit jinak. K tomu se využívá dekorátor `@login_required`.

Ukázka kódu – odhlášení uživatele:

```
@app.route('/odhlaseni')
@login_required
def odhlaseni():
    logout_user()
    return redirect('/prihlaseni')
```

K tomu, aby bylo možné zkontrolovat, zdali se přihlašuje existující uživatel, jsou zadaná data do přihlašovacího formuláře porovnávána s daty uloženými v databázi.

Ukázka kódu – validace při přihlašování:

```
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>'
```

3.3.4 Přidání administrátorského uživatele

Způsoby pro přidání uživatele, který bude mít administrátorská práva, jsou dva. První z nich je ten, že uživatel s již aktivním administrátorským účtem vytvoří skrze administracní prostředí na adrese `/novy-uzivatel` nového účastníka. Ten automaticky dostane práva administrátora.

Druhou možností je použít příkazu na serveru, na kterém je aplikace spuštěna. Příkaz tak následně pomocí předaných parametrů uživatele s administrátorským oprávněním vytvoří.

Ukázka kódu – vytvoření příkazu pro přidání administrátorského uživatele:

```
@app.cli.command("create_superuser")
@click.option("--username", "-u", help="Jmeno uzivatele", required=True, prompt="Jmeno uzivatele")
@click.option("--password", "-p", help="Heslo uzivatele", required=True, prompt="Heslo uzivatele (8 znaku a vice)")
def new_user(username, password):
    if len(username) > 0 and len(password) > 7:
        hashed_password = generate_password_hash(password, method='sha256')
        new_user = Uzivatel(jmeno=username, heslo=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        print("Uzivatel vytvoren")
    else:
        print("Neplatne zadani")
```

Tři možné užití téhož příkazu, jímž lze vytvořit uživatele s administrátorskými právy:

```
$ docker-compose exec backend flask create_superuser
$ docker-compose exec backend flask create_superuser -u <jmeno> -p <heslo>
$ docker-compose exec backend flask create_superuser --username <jmeno>
--password <heslo>
```

3.4 Automatizace obrázků

3.4.1 Generování obrázků pro sociální síť

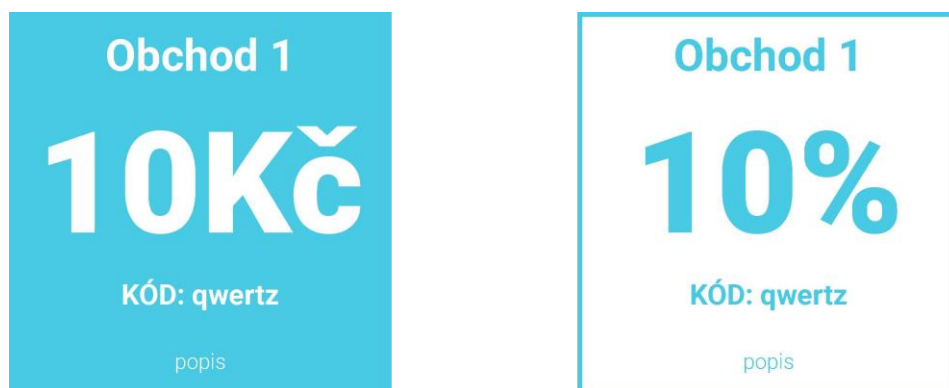
Generování obrázků je z velké části zrealizováno pomocí knihovny Pillow 8.4.0. Tato knihovna umožňuje vytvořit nový obrázek, psát na něj text a následně upravený obrázek uložit.

Jednotlivá data, jako je například název obchodu, slevový kupón a výše slevy, jsou funkci `generate_image()` předávána jako argumenty, které jsou poté dosazeny na obrázek ve formě textu.

Ukázka kódu – vytvoření textu týkající se výše slevy z dosazených hodnot:

```
font = ImageFont.truetype(FONT_BLACK, 400)
w, h = font.getsize(f"{sleva}{typ}")
draw = ImageDraw.Draw(bg)
draw.fontmode = "1"
draw.text(((1080-w)/2, DISCOUNT_H), f"{sleva}{typ}", font=font, fill=curr_color)
```

Aby vygenerované obrázky nebyly jednobarevné, střídají se typy pozadí, viz ukázky. Typ následujícího obrázku je uložen v souboru *colorset.txt* v podobě čísla. Číslo 1 značí modré pozadí s bílým textem, naopak číslo 2 znamená bílé pozadí a modrý text.



Obrázek 12: Porovnání variant vygenerovaných obrázků

3.4.2 Generování obrázků z administračního prostředí

Pokud se v administračním prostředí vytvoří či upraví kupón typu *procenta* nebo *koruny*, vygeneruje se automaticky obrázek s daty pomocí funkce `create_img()` a následně se uloží do složky *web/backend/app/create_img/img*.

Ukázka kódu – generování obrázků typu koruny z administračního prostředí:

```
class KuponView(ModelView):
    def after_model_change(self, form, model, is_created):
        if(model.typ == Typ.koruny):
            obchod = Obchod.query.get(model.obchod_id)
            create_img(obchod.nazev, model.kod, model.popis, model.sleva, "Kč")
```

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 síť.

Rozpracované cíle:

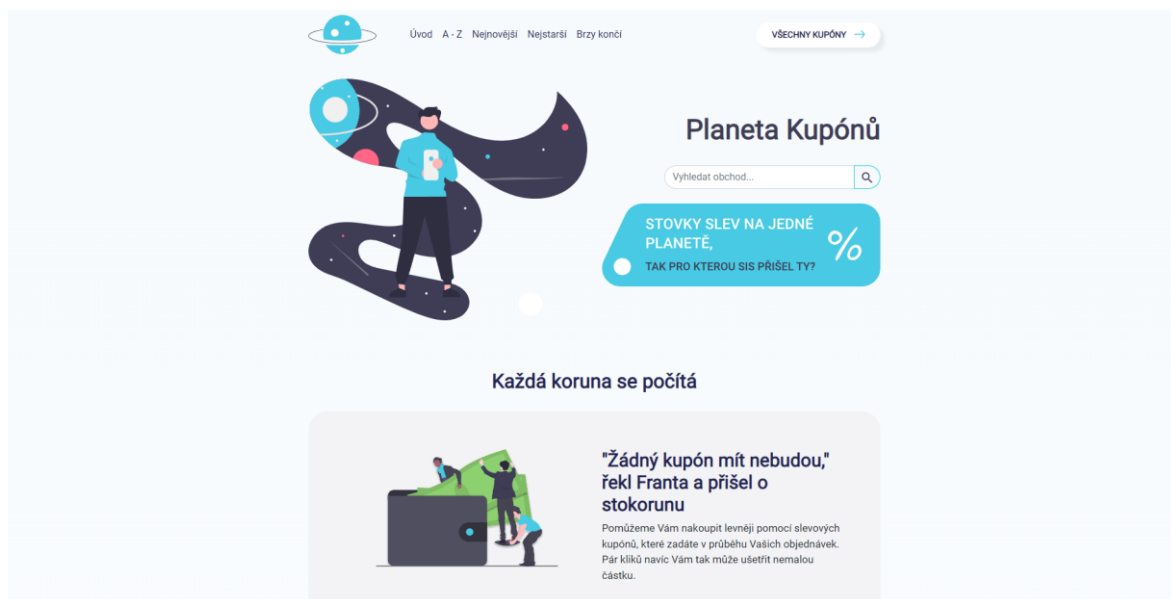
- Vytvoření skriptu, který by sbíral volně dostupné kódy a automaticky je ukládal do databáze,
- desktopová aplikace pro zobrazování kuponů.

4.2 Uživatelský manuál

4.2.1 Běžný uživatel

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

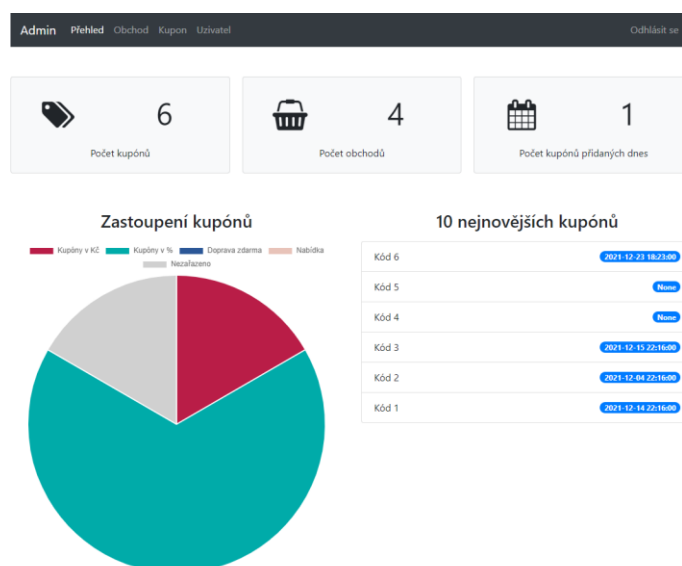
Níže na úvodní stránce se nachází informační banner a přehled 9 nejnovějších kuponů.



Obrázek 13: 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 14: Zobrazení pro administrátora

ZÁVĚR

Cílem projektu bylo vytvořit plně funkční webovou aplikaci, která by shromažďovala slevové kupóny.

Backend webové aplikace byl postaven na mikro frameworku Flask 2.0.1 a následně rozšířen o REST API pomocí Flask-RESTful 0.3.9 a také o administrační prostředí, které je zajištěno rozšířením Flask-Admin 1.5.8. Frontend je zpracován javascriptovou knihovnou React.js 17.0.2. Ten následně získává data pro zpracování zasíláním dotazů na příslušné endpointy API.

Stěžejní cíle byly splněny, projekt je funkční, kupóny jdou bez problému přidávat, avšak pouze ručně. Zde se nabízí první možnost budoucího vylepšení – automatické přidávání kupónů. Aplikace by tak byla více zautomatizovaná a byla by menší potřeba ručních zásahů do dat.

Dalším vylepšením by mohla být mobilní aplikace, jelikož v dnešní době stále více lidí hledá informace na mobilních telefonech.

Věřím, že projekt má možnost se uplatnit v praxi, zvláště v České republice, kde lidi při nakupování převážně zajímá sleva.

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

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

- [1] Bootstrap. *Build fast, responsive sites with Bootstrap* [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](https://realpython.com/flask-connexion-rest-api/) [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](https://flask-restful.readthedocs.io/en/latest/). [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](https://www.javascripttutorial.net/javascript-fetch-api/). [cit. 2021-12-30]. Dostupné z <https://www.javascripttutorial.net/javascript-fetch-api/>.
- [7] MDN Web Docs. *Using Fetch* [online], [developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch). 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.