Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÁ PRÁCE

2023/2024

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÁ PRÁCE

Předmět: Programování

Téma: Bazar

Autoři: Jan Ševčík, Radmin Tříletý, Vojtěch Kadlec

Třída: 3. E

Školní rok: 2023/2024

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V …….. dne …… ……
Jan Ševčík

V …….. dne …… ……
Radim Tříletý

V …….. dne …… ……
Kadlec Vojtěch

Anotace

Toto je dokumentace naší ročníkové práce. V této práci jsou popsány detaily a funkce našeho výsledného produktu. Zadáním našeho ročníkového projektu bylo vytvořit webovou aplikaci, která bude využívána jako bazar. Začátkem školního roku 2023/24 jsme si vybrali toto zadání a snažili jsme se o její zkompletování. Podstatou této aplikace je, že uživatel si otevře webovou stránku a bude si moci procházet příspěvky od ostatních uživatelů. Vždy bude mít k dispozici název, popis produktu, cenu a obrázky. Následně po otevření inzerátu se zde objeví i kontaktní údaje na danou osobu. S tou se pak můžete pohodlně online domluvit, ať už za pomocí emailu, telefonního čísla, nebo také pomocí chatu, který je v naší webové aplikaci zabudován, který slouží ke zlepšení a zpohodlnění komunikace mezi uživateli. Jako uživatel také můžete jednoduše založit nabídku a inzerovat jí na naší stránce. Cílem našeho ročníkového projektu, bylo zdokonalit naše zadání tak, aby lidé mohli obchodovat, přes náš web bezproblémově a z pohodlí jejich domova.

Subtract

This is documentation of our year's project. This work describes the details and features of our final product. The assignment of our year project was to create a web application that will be used as a bazaar. At the beginning of the 2023/24 school year, we chose this assignment and worked to make it complete. The idea behind this app is that the user will open a web page and be able to browse through the contributions from other users. The name, product description, price and pictures will always be available. Subsequently, when the advert is opened, the contact details of the person will also appear. You can then conveniently talk to the person online, either by email, phone number, or by using the chat feature built into our web application, which is designed to improve and facilitate communication between users. As a user, you can also easily create an offer and advertise it on our site. The aim of our year-long project was to improve our assignment so that people can trade, through our website seamlessly and from the comfort of their home.

Zadání

Ročníkový projekt bude webová aplikace, která bude obsahovat internetový bazar. Uživatel bude mít možnost založit si účet a následně přidat vlastní inzeráty, jak na koupi tak na prodej daného produktu. Budou zde jednotlivé sekce (př.: automobily, elektronika, atd.). U každého inzerátu bude možnost přidat cenu, popis, foto a kontaktní údaje. Kupující bude mít možnost vyhledávat podle různých kritérií (kategorie, názvu, …) a zprostředkuje prodej. Aplikace bude mít hodnotící systém uživatelů.

Obsah

1 Úvod **1**
2 Frontend **2**
2.1 Struktura Frontend části 2
2.2 homepage 3
2.2.1 home.html 3
2.2.2 login.html 3
2.2.3 offer.html 4
2.2.4 signup.html 4
2.2.5 reset_password.html 4
2.3 profilepage 5
2.3.1 personal_info.html 5
2.3.2 user_offers.html 5
2.3.3 your_order_detail.html 5
2.3.4 payment_confirmed.html a payment_canceled.html 6
2.3.5 payment_confirmation.html 6
2.4 chat 7
2.4.1 chat.html 7
3 Backend **8**
3.1 Struktura projektu 8
3.2 Jednotlivé soubory 8
3.2.1 models.py 8
3.2.2 urls.py 9
3.2.3 views.py 9
3.2.4 forms.py 10
3.3 Bazar 11
3.3.1 settings.py 11
3.3.2 urls.py 16
3.4 homepage 18
3.4.1 models.py 18
3.4.2 urls.py 21
3.4.3 forms.py 23
3.4.4 views.py 25
3.5 profilepage 31
3.5.1 urls.py 31
3.5.2 forms.py 32
3.5.3 views.py 33
3.6 Chat 37
3.6.2 urls.py 37
3.6.3 views.py 37
4 Použité Api **38**
4.1 Google OAuth 38
4.2 Paypal Api 39

4.3 Geoapify Api 39
5 Nasazení projektu 40
6 Závěr 41
Použitá literatura 42
Seznam obrázků 44

1 Úvod

Tento spis je dokumentací našeho ročníkového projektu. Tématem našeho ročníkového projektu je programování webové aplikace “Bazaros”. Tato dokumentace shrnuje informace o našem vytvořeném programu. Zdokumentovali jsme většinu kódu, popsali jednotlivé kroky a jednotlivé funkce. V této dokumentaci najdeme vše od designu této webové aplikace, přes její fungování. Zároveň u vybraných částí popisujeme problémy, které v programování nastaly a jak jsem je vyřešil. Celý program byl naprogramován v jazyce Hypertext Markup Language, Cascading Style Sheets, Javascript a Python.

2 Frontend

2.1 Struktura Frontend části

Naše webová aplikace je rozdělena do čtyř hlavních částí, každá s vlastním HTML, CSS a javascriptem. V základní struktuře se nachází homepage, profile page a chat. Pod každým z těchto odvětví se ukrývá několik souborů s frontend obsahem.

Homepage**:** Tato část je první stránkou, kterou uživatel po vstupu na stránku vidí. Obsahuje hlavní stranu naší aplikace, odkud jsou všechny funkce jednoduše přístupné a dohledatelné.

Profilepage: V této části nalezneme HTML pro správu uživatelských profilů a samotných objednávek. Uživatelé mohou zobrazit a upravit své osobní informace, nastavení účtu a další funkce jako například zjistit si informace o inzerátech.

Chat: Tato část umožňuje uživatelům komunikovat mezi sebou. Obsahuje okno pro psaní zpráv.

Každá z těchto částí by má vlastní HTML, CSS a JavaScript, který definuje jejich vzhled a chování. Tyto soubory jsou organizovány do samostatných složek pro každou část, což usnadňuje správu a aktualizaci kódu.

2.2 homepage

V adresáři "homepage" nalezneme celkem osm souborů HTML: "home.html", "login.html", "offer.html", "password_reset_complete.html", "password_reset_confirm.html", "reset_password.html", "reset_password_sent.html" a "signup.html". Tyto soubory jsou klíčové pro funkcionalitu naší webové aplikace nazvané Bazaros. Každý z těchto souborů se zaměřuje na různé aspekty uživatelského rozhraní, zlepšující zážitek uživatele při prohlížení nabídek a interakci s aplikací hlavně zaměřené na přihlašovací funkce.

Kromě HTML souborů máme také jeden CSS soubor nazvaný "homepage.css", který slouží k úpravě vzhledu všech stránek v aplikaci. Tento soubor zajišťuje konzistentní vizuální styl a estetiku.

Kromě toho máme také textový soubor "custom_mail.txt", který je custom email, který je poslaný, kvůli změně hesla.

Celkově tyto soubory společně tvoří základní stavební kameny pro přihlašování a koukání na inzeráty pro uživatele v aplikaci Bazaros.

2.2.1 home.html

Toto HTML má za účel, zpříjemnit uživateli čas strávený v naší aplikaci. Na této hlavní stránce najdete všechny důležité věci příjemně na jednom místě.

Na vrchní části můžete vidět šedý “navbar”, na kterém najdeme název našeho webu. Následně zde máme pár užitečných odkazů, které uživatele přesměrují přesně tam kam potřebuje. V pravé části navbaru můžeme najít vyhledávací okénko. Díky klíčovým slovům si můžeme například vyhledávat inzeráty jiných uživatelů. Následně zde najdeme ještě jedno velmi důležité tlačítko a tím je “Open Filter”, které slouží k filtrováním produktů a k usnadnění vyhledávání.

Následně na zbytku obrazovky můžeme vidět náhledy na dané inzeráty, které je možné rozkliknout a zjistit si o dané inzerci více informací.

2.2.2 login.html

V tomto HTML je účel jasný a to se přihlásit ke svému účtu. Pokud snad účet ještě nemáte, tak na stránce najdete tlačítko “Vytvořit účet” nebo pokud se nemůžete přihlásit, tak zde najdete “Nemůžete se přihlásit?” Tyto tlačítka vás přesměrují na stránku dle vašeho požadavku.

Následně na stránce najdete kolonku na vyplnění uživatelského jména a hesla, pokud jsou vaše údaje zadány správně, budete přesměrování zpět na homepage. Po zadání nesprávných údajů se vám ukáže, že někde nastala chyba. Také se do účtu máte možnost přihlásit Google účtem, přes zelené tlačítko.

2.2.3 offer.html

Tento HTML dokument obsahuje náhled a informace o uživateli a jeho inzerátu. Najdete zde kontaktní údaje a celý popis o produktu. Následně zde můžete produkt komentovat a číst si již existující komentáře. Tento dokument také obsahuje již zmíněný navbar (viz. kapitola 2.2.1).

2.2.4 signup.html

Tato stránka slouží k zakládání uživatelských účtů. Při prvním setkání s touto stránkou zjistíte, že jsou zde kolonky na vyplnění osobních údajů jako například, už. jméno, jméno, příjmení, email, heslo. Následně zde před založením účtu musíte prokázat, že nejste robot.

Pokud je vše vyplněno tak, jak má být, automaticky se vám založí účet, který budete moct následně používat. Pokud zde nastala chyba, zobrazí se červený obdélník, který vám řekne možnosti, co jste nevyplnili, nebo vyplnili ale chybně.

2.2.5 reset_password.html

Toto html se zobrazí, pokud kliknete na zmiňované tlačítko “Nemůžete se přihlásit?”. Tlačítko vás odkáže na vyskakovací okno v html, které po vás požaduje zadání emailové adresy, kterou jste používali při zakládání vašeho uživatelského účtu. Po vyplnění kliknete na tlačítko “Poslat email” a následně vám přijdou všechny potřebné instrukce do vašeho mailu.

2.3 profilepage

V adresáři "homepage" nalezneme celkem šest HTML souborů: "payment_canceled.html", "payment_confirmation.html", "payment_confirmed.html", "personal_info.html", "user_offers.html" a "your_order_detail.html". Tyto soubory společně s jedním CSS souborem, "profilepage.css", poskytují klíčovou funkčnost naší webové aplikace Bazaros.

Jsou navrženy tak, aby zlepšily uživatelský komfort a usnadnily navigaci uživatelům při provádění transakcí, správě osobních informací a prohlížení nabídek a poptávek uživatelů.

Každý z těchto HTML souborů a přidružený CSS soubor jsou pečlivě navrženy a implementovány s cílem zajistit plynulý a uživatelsky příjemný průběh používání naší webové aplikace Bazaros.

2.3.1 personal_info.html

Tento HTML dokument představuje uživatelskou stránku s osobními informacemi. Jeho struktura je navržena tak, aby poskytovala uživatelům možnost zobrazovat a upravovat své údaje. Jedním z zajímavých prvků je dynamické zobrazení formulářů pro úpravu osobních údajů pomocí JavaScriptu. Když uživatel klikne na tlačítko "Upravit" vedle určitého údaje, příslušný formulář se zobrazí a umožní mu provést úpravu. Po dokončení úprav může uživatel potvrdit změny stisknutím tlačítka "Upravit". Tato interaktivní funkcionalita přispívá k uživatelské přívětivosti a umožňuje uživatelům snadno spravovat své osobní údaje. Tento dokument také obsahuje již zmíněný navbar (viz. kapitola 2.2.1), taky zde je možné vidět vaše hodnocení od ostatních uživatelů a komentáře (pokud nějaké máte). Můžete zde také odstranit svůj účet.

2.3.2 user_offers.html

Tento HTML dokument slouží k vytvoření uživatelského rozhraní pro zobrazení inzerátů a možnost jejich editace. Samotný obsah stránky je rozdělen do dvou sloupců. V levém sloupci se zobrazují existující inzeráty uživatele. Každý inzerát je zobrazován v odděleném boxu obsahujícím název, cenu a náhledový obrázek. Barva tohoto boxu se mění podle zvoleného tématu. Pokud uživatel vybere editaci inzerátu, je přesměrován na stránku pro úpravu konkrétního inzerátu. V pravém sloupci je formulář pro vytvoření nového inzerátu. Tento formulář obsahuje pole pro název inzerátu, popis, kategorii a cenu. Barva formuláře se opět mění podle aktuálního tématu.

2.3.3 your_order_detail.html

Tento HTML dokument je určen pro úpravu existujícího inzerátu a zahrnuje formuláře pro změnu různých parametrů inzerátu. Hlavní obsah stránky je rozdělen do několika formulářů, které umožňují úpravu titulku, popisu, ceny a kategorie inzerátu. Každý formulář obsahuje odpovídající vstupní pole a tlačítko pro uložení změn. Níže je umístěn další blok s

informacemi o možnosti zviditelnění inzerátu za poplatek, včetně ceny a odkazu na nákup. Tento dokument také obsahuje již zmíněný navbar (viz. kapitola 2.2.1), který obsahuje navíc dvě tlačítka: “Smazat inzerát” a “Obnovit inzerát”.

2.3.4 payment_confirmed.html a payment_canceled.html

Tyto dva HTML dokumenty slouží k zobrazení stavu platby uživateli, a to buď potvrzení platby nebo oznámení o zrušení platby. Oba dokumenty mají podobnou strukturu a vzhled, však se liší barvou a ikonou zobrazenou vedle zprávy. Oba dokumenty jsou responzivní a používají Bootstrap framework pro konzistentní vzhled a chování.

2.3.5 payment_confirmation.html

Tento HTML dokument slouží k potvrzení nákupu a zobrazení informací o platbě, včetně ceny, způsobu platby a dalších detailů. Celkově je struktura jednoduchá, ale efektivně zobrazuje potřebné informace o nákupu a umožňuje uživateli provést platbu pomocí PayPalu.

2.4 chat

V rámci adresáře "chat" nalezneme jeden HTML soubor, pojmenovaný "chat.html", který zásadně ovlivňuje a zdokonaluje uživatelskou zkušenost naší webové aplikace Bazaros. Jeho hlavním úkolem je poskytnout uživatelům interaktivní chatovací rozhraní, které umožňuje komunikaci a vyjednávání mezi uživateli. Tento soubor je pečlivě navržen tak, aby ladil s celkovým designem aplikace. Jeho obsah se zaměřuje na prezentaci dialogů a interakcí mezi uživateli prostřednictvím textových zpráv.

2.4.1 chat.html

Tato struktura HTML dokumentu slouží k vytvoření uživatelského rozhraní pro chatování. Obsahuje tlačítko "Zpět", které umožňuje uživatelům navigovat zpět na předchozí stránku. Centrální částí chatu je rozhraní, kde jsou zobrazovány zprávy a uživatelé mohou psát nové. Zprávy jsou prezentovány ve "bublinách" s barevným odlišením podle odesílatele. Textové pole umožňuje psát nové zprávy a tlačítko pro odeslání aktivuje odeslání zprávy.

3 Backend

3.1 Struktura projektu

Projekt se skládá ze 3 modulů djangem pojmenovaných jako aplikace. Každá z těchto aplikací se stará o jednotlivé části webové aplikace. Projekt je takto rozdělen kvůli lepší orientaci ve vytváření projektu. Projekt je vygenerován pomocí příkazu django-admin startproject <jméno projektu> a každá další aplikace je vygenerovaná příkazem python manage.py startapp <jméno aplikace>.

První částí projektu, která však není aplikací je složka Bazar (viz. kapitola 3.3), která je vygenerovaná prvním příkazem (django-admin startproject <jméno projektu>). Obsahuje dva důležité soubor „settings.py" (viz. kapitola 3.3.1), který obsahuje konfiguraci projektu a „urls.py" (viz. kapitola 3.3.2), který se stará o rozdělení různých volaných cest.

Další tři aplikace se starají každá o určitou část projektu. Aplikace homepage (viz. kapitola 3.4) se stará o úvodní stránku aplikace, která zobrazuje detaily inzerátů, stará se o přihlášení uživatele a o vyhledávání inzerátů podle různých parametrů. Druhá aplikace profilepage (viz. kapitola 3.5) se stará o upravení informací o uživateli, vytváření inzerátů a upravování inzerátů. Poslední aplikace chat (viz. kapitola 3.6) je vytvořena pro zprostředkování komunikace mezi dvěma uživateli.

3.2 Jednotlivé soubory

3.2.1 models.py

Obsahuje veškerou konfiguraci databáze sepsanou v jednotlivých třídách. Každá třída v tomto souboru je jedna tabulka v databázi. Každý parametr jako jeden sloupec tabulky. Aby bylo možné pro django převést tyto třídy do sql databáze je potřeba ještě definovat jakého druhu sloupce jsou (varchar, int, …). Tohoto je docíleno pomocí třídy models, jejichž metody jsou volány a vrácené hodnoty přiřazené jednotlivým parametrům.

```python
from django.db import models


class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=100)
    publication_date = models.DateField()

    def __str__(self):
        return self.title
```

obr. 1: Příklad třídy v souboru „models.py”

Pro propsání tohoto souboru do databáze je potřeba použít dvou příkazů. První z nich je [python manage.py makemigrations], který vytvoří soubory obsahující změny databáze podle souboru „models.py”. Příkaz [python manage.py migrate] poté provede změny v souborech vytvořených předchozím příkazem v databázi.

3.2.2 urls.py

Všechny soubory „urls.py” obsahují konfiguraci jednotlivých cest v aplikaci. Starají se o to, aby byla zavolána správná metoda v souboru „views.py” (viz. kapitola 3.2.3). Všechny cesty jsou zde uvedeny v poli s názvem „urlpatterns” a jsou vytvářeny pomocí metody path. První parametr potřebný pro metodu path je string obsahující cestu zobrazenou v prohlížeči, druhý parametr je metoda, která má být touto cestou zavolána a poslední je jméno této cesty, které se dále používá v kódu.

3.2.3 views.py

„views.py” obsahuje jednotlivé metody starající se o jednotlivé cesty v aplikaci. Každá metoda, která je volána cestou, musí vrátit metodu „redirect”, která přesměruje uživatele, nebo metodu „render”, které uživateli zobrazí příslušný html soubor. Metoda „redirect” potřebuje pouze jeden parametr, a to sice string, který obsahuje cestu, na kterou bude uživatel přesměrován. Metoda „render” bere jako parametry pole request, cestu k html souboru a kontext, který je pythonový objekt typu „dictionary”, obsahující informace, které jsou později zobrazené v html souboru.

V jednotlivých metodách může být poté uděláno cokoliv. Může být měněna databáze, získávány informace z databáze, dotazovány api a podobně. V těchto metodách je obecně celá logika všeho, co se v backendu serveru odehrává.

3.2.4 forms.py

Tento soubor je využíván k vytváření formulářů, které mohou být zobrazeny v jednotlivých html souborech. Uživatel je vyplní a data z nich poté mohou být zpracována. Jednotlivé formuláře jsou podobně jako modely v souboru definovány třídou a jednotlivé jejich části poté parametry. Opět podobně jako u „models.py" (viz. kapitola 3.2.1) jsou parametrů připisované hodnoty, které vrací jednotlivé funkce třídy „forms" (vestavěné do djanga).

3.3 Bazar

3.3.1 settings.py

Tato část celé webové aplikace je z větší části vygenerovaná djangem, ale pro funkčnost aplikace bylo potřeba přidat několik řádků kódu a upravit některé její části. První upravenou částí bylo přidání souboru „config.json”, tento soubor obsahuje citlivé údaje o projektu, které by neměly být přístupné veřejnosti, tento soubor je načtený a je používán napříč kódem (Schafer, 2018).

```
import json

with open("config.json") as config_file:
    config = json.load(config_file)

# Příklad použití konfiguračního souboru
SECRET_KEY = config["SECRET_KEY"]
```

obr. 2: Načtení souboru „config.json” a jeho použití

Další upravenou částí musí být hodnota „ALLOWED_HOSTS” zde musela být přidána ip adresa projektu a případně doména, aby projekt přijímal requesty přicházející z těchto adres. Tyto adresy jsou „domovprojekt.com” a veřejná ip adresa serveru.

Pro správné zacházení projektu s obrázky a videi a statickými soubory jako jsou soubory „.css” a „.js” soubory. Museli být upraveny hodnoty proměnných „MEDIA_ROOT” a „MEDIA_URL” pro obrázky a videa a „STATIC_ROOT” a „STATIC_URL” pro statické soubory.

```
import os

MEDIA_ROOT = os.path.join(BASE_DIR, "media")
MEDIA_URL = "/media/"

STATIC_ROOT = os.path.join(BASE_DIR, "static")
STATIC_URL = "/static/"
```

obr. 3: Konfigurace cest k statickým a media souborům

Pro správnou funkčnost přihlašování a odhlašování uživatelů museli být přidány proměnné „LOGIN_URL”, „LOGIN_REDIRECT_URL”, „LOGOUT_REDIRECT_URL”. Tyto proměnné určují, kde je stránka pro přihlášení a kam bude uživatel přesměrován po přihlášení nebo odhlášení.

```
LOGIN_URL = "/login/"
LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"
```

obr. 4: Konfigurace cest k přihlášení

Pro správné fungování různých aplikací obsahuje soubor „settings.py” pole „INSTALLED_APPS”. V tomto poli musí být přidány k djangem předem přidaným aplikacím přidány i všechny vytvořené aplikace jako v případě tohoto projektu homepage, profilepage a chat.

Toto pole však neobsahuje pouze vytvořené aplikace, ale i nainstalované aplikace do virtuálního prostředí. Tyto aplikace jsou komentářemy v kódu rozděleny do kategorií. Jsou zde aplikace potřebné pro funkčnost přihlašování do aplikace pomocí googlu, které jsou v kódu pod komentářem „apps needed for google login”. (Akamai DevRel, 2022) Pod dalším komentářem „captcha” je nainstalovaná aplikace, která je využívána při přihlašování uživatelů pro ověření, zda nejsou pouze robot. (Ivy, 2021) Jelikož náš projekt obsahuje možnost koupit si propagaci inzerátu je tu nainstalovaná pod komentářem „paypal” i aplikace potřebná k funkčnosti této části projektu. (THE PROTON GUY, 2023) Poslední části jsou už jen ostatní nainstalované aplikace používané pro různé menší části projektu pod komentářem „other apps”.

```
INSTALLED_APPS = [
    # default
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    # all the created apps
    "homepage",
    "profilepage",
    "chat",
    # other apps
    "widget_tweaks",
    "ckeditor",
    "ckeditor_uploader",
    # apps needed for google login
    "django.contrib.sites",
    "allauth",
    "allauth.account",
    "allauth.socialaccount",
    "allauth.socialaccount.providers.google",
    # captcha
    "captcha",
    # paypal
    "paypal.standard.ipn",
]
```

obr. 5: Pole „INSTALLED_APPS" obsahující nainstalované aplikace

Část, která následuje je potřebná pro funkčnost přihlášení pomocí google účtu. Je jí pole „SOCIALACCOUNT_PROVIDERS" proměnná „SOCIALACCOUNT_LOGIN_ON_GET", pole „AUTHENTICATION_BACKENDS" a dalo by se sem počítat i djangem předem vygenerované pole „MIDDLEWARE" do kterého ale musel být přidán řádek a to sice „allauth.account.middleware.AccountMiddleware". Pole „SOCIALACCOUNT_PROVIDERS" muselo být společně s proměnou „SOCIALACCOUNT_LOGIN_ON_GET" a polem „AUTHENTICATION_BACKENDS" vytvořeno a přidán do nich následující kód. (Akamai DevRel, 2022)

```
SOCIALACCOUNT_PROVIDERS = {
    "google": {
        "SCOPE": ["profile", "email"],
        "AUTH_PARAMS": {"access_type": "online"},
    }
}
SOCIALACCOUNT_LOGIN_ON_GET = True

AUTHENTICATION_BACKENDS = (
    "django.contrib.auth.backends.ModelBackend",
    "allauth.account.auth_backends.AuthenticationBackend",
)
```

obr. 6: Konfigurace potřebná k správné funkčnosti přihlašování

Pole „DATABASES" muselo být také upraveno, protože náš projekt nepoužívá djangem v základu používanou databázi SQLite, ale databázi postgresSQL, aby tedy propojení správně fungovalo musel být nainstalován do virtuálního prostředí modul psycopg2, do pole přidán následující kód a přihlašovací údaje uložené v souboru „config.json". (stackpython, 2020)

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql_psycopg2",
        "NAME": config["DATABASE_NAME"],
        "USER": config["DATABASE_USER"],
        "PASSWORD": config["DATABASE_USER_PASSWORD"],
        "HOST": "127.0.0.1",
        "PORT": "5432",
    }
}
```

obr. 7: Konfigurace databáze

Náš projekt používá pro umožnění uživateli vytváření popisů jednotlivých inzerátu program ckeditor. (Uddin, 2022) Pro jeho správnou funkčnost muselo do „settings.py" přidáno pole „CKEDITOR_CONFIGS", které určuje, jaké různé funkčnosti může editor mít. (Wazy, 2021) Poslední řádkou přidanou kvůli funkčnosti aplikace je proměná „CKEDITOR_UPLOAD_PATH". (Smudja, 2020)

```
CKEDITOR_CONFIGS = {
    "default": {
        "skin": "n1theme",
        "toolbar": "Custom",
        "toolbar_Custom": [
            ["Format", "Font", "FontSize"],
            ["Bold", "Italic", "Underline"],
            [
                "NumberedList",
                "BulletedList",
                "-",
                "Outdent",
                "Indent",
                "-",
                "JustifyLeft",
                "JustifyCenter",
                "JustifyRight",
                "JustifyBlock",
            ],
            ["Cut", "Copy", "Paste", "PasteText"],
            ["Image"],
        ],
        "extraPlugins": "image2",
        "filebrowserBrowseUrl": "",
    }
}

CKEDITOR_UPLOAD_PATH = "uploads/"
```

obr. 8: Konfigurace aplikace ckeditor

Kvůli správné konfiguraci paypalu museli do souboru „settings.py" být přidány tři proměné, kterými jsou „PAYPAL_RECIEVER_EMAIL", „PAYPAL_TEST" a „PAYPAL_BUY_BUTTON_IMAGE". „PAYPAL_RECIEVER_EMAIL" určuje, kdo přijme platbu, „PAYPAL_TEST" určuje, zda je integrace ve vývoji a v poslední řadě „PAYPAL_BUY_BUTTON_IMAGE" je pouze cesta k obrázku používanému jako tlačítko k nákupu. (THE PROTON GUY, 2023)

```
PAYPAL_RECEIVER_EMAIL = config["PAYPAL_EMAIL"]
PAYPAL_TEST = True
PAYPAL_BUY_BUTTON_IMAGE = "/media/images/paypal.jpg"
```

obr. 9: Konfigurace aplikace paypal

Pro to aby projekt měl možnost posílat emaily na změnu hesla muselo být změněno několik proměnných. Tyto proměnné jsou „EMAIL_BACKEND”, „EMAIL_PORT”, určuje port, který používá provozovatel smtp serverů na posílání emailů, „EMAIL_HOST”, která obsahuje doménu smtp serverů poskytovatele emailu, dvojice „EMAIL_USE_TLS” a „EMAIL_USE_SSL”, který protokol má projekt použít a poslední dvojice je „EMAIL_HOST_USER” a „EMAIL_HOST_PASSWORD”, které obsahují emailovou adresu a heslo k ní (tyto používá projekt pro posílání emailů).

```
EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
EMAIL_PORT = 587
EMAIL_HOST = "smtp.seznam.cz"
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
EMAIL_HOST_USER = config["EMAIL_HOST_USER"]
EMAIL_HOST_PASSWORD = config["EMAIL_HOST_PASSWORD"]
```

obr. 10: Konfigurace potřebná pro posílání emailů

Poslední upravenou částí „settings.py” je pole „CSRF_TRUSTED_ORIGINS”, které určuje ze, kterých adres mohou přijít do aplikace takzvané csrf tokeny. Tyto adresy jsou nastavené na „http://192.168.88.22” a „http://domovprojekt.com”.

3.3.2 urls.py

Při prvním vygenerování obsahuje tento projekt pouze pole „urlpatterns” a v něm jedinou takzvanou „cestu”, která zařizuje funkčnost stránky pro uživatele, kteří spravují stránku.

Další tři cesty, které musely být do pole „urlpatterns” jsou cesty k třem různým aplikacím, a to sice homepage (viz. kapitola 3.4), profilepage (viz. kapitola 3.5) a Chat (viz. kapitola 3.6). Jsou zde, aby se zavolal další správný doubor „urls.py” v ostatních aplikacích.

```
urlpatterns = [
    path("", include("homepage.urls"), name="home"),
    path("profilepage/", include("profilepage.urls"), name="profile"),
    path("chat/", include("chat.urls"), name="chat"),
]
```

obr. 11: Cesty k aplikacím

Následuje pět dalších cest zajišťujících funkčnost různých nainstalovaný komponentů aplikace. Prvním z nich odkazuje na soubor „urls.py" aplikace allauth, která je používána na registraci uživatelů pomocí googlu. (Akamai DevRel, 2022) Další se odkazuje na stejný soubor, ale aplikace captcha používané při registraci na kontrolu, zda není uživatel robot. (Ivy, 2021) Následující je soubor „urls.py" aplikace ckeditor_uploader, která zařizuje textový editor pro uživatele vytvářející inzeráty. (Smudja, 2020) Poslední z cest je cesta k „urls.py" aplikace paypal, díky které je možné zapojit do aplikace platební bránu paypal. (THE PROTON GUY, 2023)

```
urlpatterns = [
    path("accounts/", include("allauth.urls")),
    path("captcha/", include("captcha.urls")),
    path(
        "ckeditor/upload/",
        login_required(ckeditor_views.upload),
        name="ckeditor_upload",
    ),
    path(
        "ckeditor/browse/",
        never_cache(login_required(ckeditor_views.browse)),
        name="ckeditor_browse",
    ),
    path("paypal/", include("paypal.standard.ipn.urls")),
    path("ckeditor/", include("ckeditor_uploader.urls")),
]
```

obr. 12: Cesty pro různé aplikace

Poslední část změněného kódu je připojeno k poli a zařizuje správnou funkčnost nahrávání media souborů na server. (Smudja, 2020)

3.4 homepage

Aplikace homepage je aplikace, kterou uživatel vidí jako první. Zařizuje přihlašování uživatelů, registraci nových účtů, vyhledávání inzerátů, filtrování inzerátů a zobrazování detailů o inzerátech.

3.4.1 models.py

Prvním modelem je model „Category" tento model má pouze jeden parametr a to sice „name", který udává, jak se bude kategorie jmenovat.

```
class Category(models.Model):
    name = models.CharField(max_length=40)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name_plural = "Categories"
```

obr. 13: Model Kategorije

Další z modelů je model „Offer", který udává parametry inzerátu. Základní parametry, která zadává uživatel jsou „Title", který je pole znaků a udává název, „price" udávající cenu a je pouze Integer, „description", který popisuje inzerát a je pole RichTextUploadingField (je to tento objekt, aby správně fungovala aplikace ckeditor (Smudja, 2020)) a „category", který obsahuje informace o tom, do které kategorie patří. Dalšími jsou parametry zařizující správnou funkčnost částí programu. První z těchto parametrů je „creation_date"(DateTimeField) a „expired" (BooleanField), které jsou využívány na kontrolování, zda není inzerát moc starý, aby nebyly zobrazovány inzeráty zbytečně. Další parametr je „importance" (IntegerField) slouží k určení kolikrát si uživatel koupil zviditelnění inzerátu, a tedy jak moc má být inzerát zviditelněn. Poslední je „preview" (CharField), které obsahuje adresu náhledového obrázku inzerátu.

```python
class Offer(models.Model):
    Title = models.CharField(max_length=255)
    price = models.IntegerField()
    description = RichTextUploadingField(blank=True, null=True)
    creation_date = models.DateTimeField(default=now)
    expired = models.BooleanField(default=False)
    creator = models.ForeignKey(User, related_name="Offers", on_delete=models.CASCADE)
    importance = models.IntegerField(default=0)
    preview = models.CharField(max_length=255, default="")
    # every category has its Offers
    category = models.ForeignKey(
        Category, related_name="Offers", on_delete=models.CASCADE
    )

    def __str__(self):
        return self.Title

    class Meta:
        verbose_name_plural = "Offers"
```

obr. 14: Model inzerátu

Třetí z modelů je „User_attachment" sloužící jako přípojené informace k základnímu modelu vytvořeného djangem a to sice „User" modelu. Čtyři základní parametry uživatele jsou „City" (CharField), „Street"(CharField), „Postal_code"(CharField) a „phone_number" (CharField). „City" ukládá město pobytu uživatele, „Street" ulici, „Postal_code" poštovní směrovací číslo a „phone_number" telefoní číslo. Další parametry jsou „rating" (IntegerField) užívaný k ukládání hodnocení uživatele, „theme"(CharField) užívaný k ukládání nastavení „light" nebo „dark" režimu a „offer_count" (IntegerField) využívaný k ukládání toho kolik má uživatel vytvořených inzerátů. Poslední z těchto parametrů je „user" (ForeinKey), který ukládá ke kterému uživateli „User_attachment" patří.

```python
class User_attachments(models.Model):
    rating = models.IntegerField(default=0)
    City = models.CharField(max_length=255, default="")
    Street = models.CharField(max_length=255, default="")
    Postal_code = models.CharField(max_length=255, default="")
    phone_number = models.CharField(max_length=255, default="")
    theme = models.CharField(max_length=255, default="")
    offer_count = models.IntegerField(default=0)
    # every on of those has its user
    user = models.ForeignKey(
        User, related_name="User_attachments", on_delete=models.CASCADE
    )

    def __str__(self):
        return self.user.username

    class Meta:
        verbose_name_plural = "User_attachments"
```

obr. 15: Model přídavných informací k uživateli

„Rating_Relation" je další z modelů a je používán na ověřování, zda uživatel hodnotil uživatele jenom jednou. Z důvodu této funkčnosti musí mít dva parametry „rating_subject" (ForeinKey) sloužící k ukládání uživatele, který byl hodnocen a „rating_creator" (ForeinKey), který ukládá hodnotícího uživatele. Poslední přídavným parametrem je „comment" (CharField), který ukládá uživatelem vytvořený komentář.

```
class Rating_Relation(models.Model):
    # rated user
    rating_subject = models.ForeignKey(
        User, related_name="Rating_Subjects", on_delete=models.CASCADE
    )
    # user rating
    rating_creator = models.ForeignKey(
        User, related_name="Rating_Creators", on_delete=models.CASCADE
    )
    # rating user comment
    comment = models.TextField(default="")

    def __str__(self):
        return (
            self.rating_subject.username + " hodnotil " + self.rating_creator.username
        )
```

obr. 16: Model „Rating_Relation"

Díky modelu „Chat" je možné spojit dva uživatele prostřednictvým aplikace, tak že si můžou posílat správy. První dva parametry jsou „user_1" (ForeinKey) a „user_2" (ForeinKey), které určují dva uživatele, který si budou psát. Další parametr „offer_id" (IntegerField), který udává, o kterém z inzerátů si uživatelé budou psát.

```
class chat(models.Model):
    user_1 = models.ForeignKey(User, related_name="User_1", on_delete=models.CASCADE)
    user_2 = models.ForeignKey(User, related_name="User_2", on_delete=models.CASCADE)
    offer_id = models.IntegerField(default=0)

    class Meta:
        verbose_name_plural = "Chats"

    def __str__(self):
        return (
            self.user_1.username
            + " chat with "
            + self.user_2.username
            + " about "
            + str(self.offer_id)
        )
```

obr. 17: Model diskuze dvou uživatelů

Další z modelů je „message", který slouží k ukládání jednotlivých zpráv zaslaných uživateli. První z parametrů modelu je „chat" (ForeinKey), kterým je určeno, v jakém z rozhovorů byly zprávy odeslány. Parametr „message_sender" (ForeinKey) určuje kdo poslal zprávu. Poslední dva parametry jsou „message" (TextField) a „creation_date" (DateTimeField), které určují text zprávy a čas s datumem poslání zprávy.

```
class message(models.Model):
    chat = models.ForeignKey(chat, related_name="Chat", on_delete=models.CASCADE)
    message_sender = models.ForeignKey(
        User, related_name="Message_Sender", on_delete=models.CASCADE
    )
    message = models.TextField()
    creation_date = models.DateTimeField(default=now)

    class Meta:
        verbose_name_plural = "Messages"

    def __str__(self):
        return self.message_sender.username + " message"
```

obr. 18: Model zprávy

Posledním z modelů je „payment", určený k ukládání jednotlivých plateb provedených v aplikaci. První dva parametry jsou typu ForeinKey „user" a „order" používány k ukládání pro který inzerát je platba vytvořena a kdo platbu provedl. Parametr „cid"(CharField) ukládá kód používaný ke specifikaci platby. Poslední dva parametry jsou „creation_date" (DateTimeField) a „completed" (BooleanField), které určují datum a čas, kdy byl inzerát vytvořen, a jestli byla platba dokončena.

```
class payment(models.Model):
    user = models.ForeignKey(User, related_name="User", on_delete=models.CASCADE)
    order = models.ForeignKey(Offer, related_name="Offer", on_delete=models.CASCADE)
    creation_date = models.DateTimeField(default=now)
    cid = models.CharField(max_length=255, default="")
    completed = models.BooleanField(default=False)

    def __str__(self):
        return self.user.username + " payment"

    class Meta:
        verbose_name_plural = "Payments"
```

obr. 19: Model platby

3.4.2 urls.py

První část pole urlpatterns v soubortu „urls.py" volá cesty k metodám, tyto metody jsou „index" volaná bez parametru, metoda „offer" volaná s jedním parametrem a to sice „offer_id" a poslední z metod je metoda „signup" volaná opět bez parametru. Cesta „home" je volána, když je uživatel na hlavní stránce, cesta „offer" když uživatel otevře některou z nabídek a cesta „signup" když si chce uživatel vytvořit účet. Poslední cesta volá metodu do djanga vestavěné třídy „LoginView" a to sice „as_view" a jejím parametrům „template_name"dává html soubor „login.html" a „authentication_form" formulář „LoginForm" (viz. kapitola 3.4.3), který je definovaný v souboru „forms.py".

```
urlpatterns = [
    path("", views.index, name="home"),
    path("offer/<int:offer_id>/", views.offer, name="offer"),
    path("signup/", views.signup, name="signup"),
    path(
        "login/",
        auth_views.LoginView.as_view(
            template_name="homepage/login.html", authentication_form=LoginForm
        ),
        name="login",
    )
]
```

obr. 20: Základní cesty aplikace homepage

Další částí jsou všechny cesty potřeba ke správné funkčnosti změnění hesla pro uživatele. (Digital World, 2023) Jedna z těchto cesta se jménem „reset_password" je upravena, aby mohl být odeslán jiný než djangem normálně vytvořený email na změnění hesla. (Ivy, 2021) Další tři cesty se jmény „password_reset_done", „password_reset_confirm" a „password_reset_domplete". Tyto cesty volají metody „as_view" tří různých do djanga vestavěných tříd. Těmto metodám je ještě dávána do parametru html soubor, který má být zobrazen na místo normálně djangem vygenerovaným.

```
urlpatterns = [
    path(
        "reset_password/",
        views.reset_password_custom,
        name="reset_password",
    ),
    path(
        "reset_password_sent/",
        auth_views.PasswordResetDoneView.as_view(
            template_name="homepage/reset_password_sent.html"
        ),
        name="password_reset_done",
    ),
    path(
        "reset/<uidb64>/<token>/",
        auth_views.PasswordResetConfirmView.as_view(
            template_name="homepage/password_reset_confirm.html"
        ),
        name="password_reset_confirm",
    ),
    path(
        "reset_password_complete/",
        auth_views.PasswordResetCompleteView.as_view(
            template_name="homepage/password_reset_complete.html"
        ),
        name="password_reset_complete",
    )
]
```

obr. 21: Cesty potřebné pro správnou funkčnost měnění hesel

Poslední dvě cesty vedou se jmény „logout” a „theme” vedou k metodám „out” a „theme”. Tyto metody jsou využívány k jednoduchým operacím. Jako je měnění režimu jasu, nebo odhlášení.

```
urlpatterns = [
    path("logout/", views.out, name="logout"),
    path("theme/", views.theme, name="theme"),
]
```

obr. 22: Ostatní cesty souboru

3.4.3 forms.py

Prvním z formulářů vytvořených v tomto souboru „forms.py” je „LoginForm” a obsahuje pouze dva parametry. Tyto parametry jsou „username” (CharField) a „password” (CharField). Oba tyto parametry mají nastavený „placeholder” (obsah který bude ve vyplňovacím html zobrazen předtím, než do něj uživatel něco zadá). (Codemy.com, 2021)

```
class LoginForm(AuthenticationForm):

    username = forms.CharField(
        widget=forms.TextInput(
            attrs={
                "placeholder": "Jmeno přes které vás oslovujeme",
            }
        )
    )
    password = forms.CharField(
        widget=forms.PasswordInput(
            attrs={
                "placeholder": "Vaše bezpečné heslo",
            }
        )
    )
```

obr. 23: Formulář pro přihlášení

Další z formulářů je „SignupForm”. První z parametrů tohoto formuláře je „captcha” obsahující formulář „CaptchaField”, který je do souboru naimportovaný. (Ivy, 2021) Dalšími parametry jsou „username”, „email”, „password1” a „password2”, které jsou všechny CharField a mají nastavený „placeholder”.

```
class SignupForm(UserCreationForm):
    captcha = CaptchaField()

    class Meta:
        model = User
        fields = (
            "email",
            "username",
            "password1",
            "password2",
            "first_name",
            "last_name",
        )

    username = forms.CharField(
        widget=forms.TextInput(
            attrs={
                "placeholder": "Jmeno uzivatele aneb jmeno, kterým vás budeme oslovovat",
            }
        )
    )
    email = forms.CharField(
        widget=forms.EmailInput(
            attrs={
                "placeholder": "Emailová adresa přes kterou vás budeme kontaktovat",
            }
        )
    )
    password1 = forms.CharField(
        widget=forms.PasswordInput(
            attrs={
                "placeholder": "Heslo",
            }
        )
    )
    password2 = forms.CharField(
        widget=forms.TextInput(
            attrs={
                "placeholder": "Zopakujte heslo",
            }
        )
    )
```

obr. 24: Formulář pro zaregistrování

Posledním z formulářů je formulář „rate" obsahující pouze dva parametry. Parametr „rating" (IntegerField), který má nastavenou maximální hodnotu na 10 a minimální na 0 a parametr „comment" (CharField), který má nastavený „placeholder".

```
class rate(forms.Form):
    rating = forms.IntegerField(min_value=0, max_value=10)
    comment = forms.CharField(
        widget=forms.Textarea(
            attrs={
                "placeholder": "Komentář",
            }
        )
    )
```

obr. 25: Formulář pro hodnocení

3.4.4 views.py

První metodou souboru „views.py" je metoda „index". Metoda nejprve zkontroluje, zda je uživatel přihlášen a neexistuje model „User_attachments" (viz. kapitola 3.4.1) příslušící tomuto uživateli a pokud tato podmínka projde vytvoří program nový model „User_attachments" (viz. kapitola 3.4.1) s parametrem „user" nastaveným na model „User", vestavěného do djanga, aktuálního uživatele. Další částí kódu je podmínka, která kontroluje, zda je uživatel přihlášen a zda existuje model „User_attachments" (viz. kapitola 3.4.1). Pokud tato podmínka projde je do proměnné „color" nastavena hodnota parametru „theme" modelu „User_attachments" (viz. kapitola 3.4.1) příslušící uživateli a pokud neprojde je do ní nastavena hodnota „white" (string). Program dále další podmínkou zkontroluje, zda metoda „POST" obsahuje „search" (string) a pokud ano do pole „offers" načte všechny modely „Offer" (viz. kapitola 3.4.1) u kterých parametr „Title" obsahuje string, který je načtený z requestu pod jménem „searched_text". Následovně program v podmínce ještě zkontroluje, zda je pole dlouhé 0 a pokud ano nastaví do proměnné „offers" string „failed". Poslední podmínka zkontroluje, zda je uživatel přihlášen a pokud ano načte do proměnné „att" model „User_attachments" (viz. kapitola 3.4.1) příslušící uživateli. Následovně program zkontroluje, zda jeden z parametrů načteného modelu „User_attachments" (viz. kapitola 3.4.1), „City", „Street", nebo „Postal_code" obsahuje prázdný string. Pokud tato podmínka projde je do proměnné „display_directions" nastavena hodnota „display" (string) a pokud ne hodnota „dont-display" (string). Poslední částí této metody je vytvoření kontextu první částí je pole „offers_imp", které je vytvořeno zapomocí zavolání metody importance s parametrem pole „offers". Další z částí kontextu jsou proměnné „offers" a „display_directions", které jsou vytvořené v předešlých částích kódu. Poslední částí kontextu je proměnná „category", který obsahuje všechny modely „Category" (viz. kapitola 3.4.1). Dále už jen metoda zavolá metodu „render" s parametry request, „homepage/home.html" (string) a s vytvořeným kontextem.

Další metodou je metoda „offer," která bere jako parametr kromě requestu proměnnou „offer_id". Nejprve program načte do proměnné „offer" model „Offer" (viz. kapitola 3.4.1) u které je parametr „id" roven proměnné „offer_id". Program následovně zkontroluje, zda není uživatel anonymní a zda existuje model „User_attachments" (viz. kapitola 3.4.1), který má parametr „user" roven modelu „User" aktuálního uživatele. Pokud tato podmínka projde je do

proměnné „color" nastavena hodnota parametru „theme" modelu „User_attachments" (viz. kapitola 3.4.1) příslušící aktuálnímu uživateli, ale pokud neprojde je do proměnné „color" nastaven string „white". Další část programu je další podmínka, která nejprve zkontroluje, zda request obsahuje metodu „POST". Když podmínka projde je do proměnné „u" načten vyplněný formulář „rate" (viz. kapitola 3.4.3) z requestu. Další podmínka v této podmínce zkontroluje, zda je načtený formulář validní a pokud ano nastaví do proměnné „rated" to co vrací metoda „userRating" s parametry „offer_id", parametrem „rating" a „comment" načteného formuláře a modelem „User" aktuálního uživatele. Poslední části této metody je už pouze vytvoření kontextu. První částí je proměnná „relation", která je vytvořena za pomocí podmínky, která zjistí, zda existuje model „Rating_relation" (viz. kapitola 3.4.1) u kterého je parametr „rating_subject" roven parametru „creator" modelu „Offer" (viz. kapitola 3.4.1) načteného do proměnné „offer", nebo parametr „rating_creator" roven modelu aktuálního uživatele a pokud ano je do proměnné místo hodnoty False dána hodnota True. Další části kontextu jsou proměnná „color" nastavená výše v metodě, formulář „rate" (viz. kapitola 3.4.3) nastavený do proměnné „u" výše v metodě, proměnná „offer" nastavené výše v metodě, proměnná „rated" také nastavené výše v metodě, proměnná „comments" obsahující všechny modely „Rating_Relation" (viz. kapitola 3.4.1) u kterých je parametr „rating_subject" roven parametru „creator" modelu „Offer" (viz. kapitola 3.4.1) načteného do proměnné „offer". Poslední částí kontextu je model „User_attachments" (viz. kapitola 3.4.1), který má jako parametr „user" nastavený model „User", který je uložený v parametru „creator" předem načteného modelu „Offer" (viz. kapitola 3.4.1) do proměnné „offer". Nakonec metoda vrátí metodu „render" s parametry request, stringem „homepage/offer.html" a kontextem.

Další dvě metody jsou „out" a „signup". Metoda „out" je jednoduchá metoda, která pouze zavolá do djanga vestavěnou metodu logout s parametrem „request" a následovně vrátí metodu „redirect" s parametrem „/". Složitější metoda signup nejprve zkontroluje, zda request obsahuje metodu „POST" a pokud ji obsahuje načte program do proměnné „u" formulář „SignupForm" (viz. kapitola 3.4.3). Následovně program zkontroluje, pokud je formulář načtený do proměnné validní a pokud ano uloží formulář a vrátí metodu „redirect" s parametrem „/login/". Pokud request však neobsahuje metodu „POST" program do proměnné „u" načte formulář „SignupForm" (viz. kapitola 3.4.3). Nakonec metoda vrátí metodu render s parametry request, stringem „homepage/signup.html" a s kontextem, který obsahuje pouze formulář načtený do proměnné „u".

Poslední metodou, která vrací metodu „render", je metoda „reset_password_custom". Nejprve metoda načte soubor „config.json" do pythonového objektu „dictionary" do proměnné „config". Následovně metoda zkontroluje, zda metoda requestu obsahuje „POST". Pokud tato podmínka projde je do proměnné „password_form" načten formulář „PasswordResetForm" vestavěný do djanga, který je vyplněný. Poté program zkontroluje, zda je načtený formulář validní a pokud je, je do proměnné „data" načten parametr „email" načteného formuláře do proměnné „password_form" a do pole „user_email" je za pomocí metody Q vestavěné do djanga načten model „User" uživatele, kterého parametr „email" je roven proměnné „data". Dále metoda zkontroluje, zda není pole „user_email" prázdné a pokud není začne iterovat tímto polem. Nejprve v tomto iterování vytvoří metoda proměnou

„subject” a nastaví do ní string „Změna hesla” a proměnou „email_template_name” a nastaví do ní string „homepage/custom_mail.txt”. Metoda poté vytvoří pythonový objekt „dictionary” do proměnné „parameters”, který obsahuje dvojice „email” a parametr „email” modelu „User” načteného v předchozí části kódu do proměnné „user”, dvojici „domain” a string „domovprojekt.com”, „site_name” a string „Bazaroos”, „uid” a string zašifrovaných bajtů parametru „id” modelu „User” předem načtený do proměnné „user”. Další dvojice jsou „token” a to co vrací metoda „make_token” importované do djanga vestavěné třídy „default_token_generator” s parametrem „user”, „protocol” a string „https” a poslední dvojicí je „user” a model načtený do proměnné „user”. Dále program za pomocí do djanga vestavěné metody „render_to_string” načte do proměnné „email” to co metoda vrací s parametry proměnné „email_template_name” a pythonové „dictionary” „parameters”. Následovně kód volá do djanga vestavěnou metodu „send_mail” s parametry „subject”, „email”, částí na začátku metody načteného souboru „config.json” pod klíčem „EMAIL_HOST_USER”, s polem obsahujícím parametr „email” modelu načteného do proměnné „user” a parametrem „fail_silently” nataveným na False. Po skončení iterování polem „user_email” už pouze program vrátí metodu „redirect” s jedním parametrem, a to sice stringem „/reset_password_sent/”. Pokud však první z podmínek neprošla je pouze vytvořen kontext obsahující formulář „PasswordResetForm”, který je vestavěn do djanga, a je vrácena metoda „render” s parametry request, stringem „homepage/reset_password.html” a kontextem. (KenBroTech, 2021)

Metoda „theme” je první metodou, která nevrací metodu „render”. Metoda bere jako parametr request. První částí metody je vytvoření proměnné „color” a nastavení jí na hodnotu parametru „color” získaného z requestu. Dále metoda zkontroluje, zda je proměnná nastavena na hodnotu „light” (string). Pokud ano zkontroluje, zda existuje model „User_attachments” (viz. kapitola 3.4.1) příslušící aktuálnímu uživateli. Pokud existuje načte ho a nastaví hodnotu jeho parametru „theme” na string „white” a uloží ho. Pokud neexistuje vytvoří model „User_atachments” (viz. kapitola 3.4.1), do parametru „user” nastaví model „User_atachments” (viz. kapitola 3.4.1) aktuálního uživatele a do parametru „theme” nastaví hodnotu „white” a následovně model uloží. Metoda dále zkontroluje, zda se proměnná „color” rovná stringu „dark” a pokud ano udělá to stejné jako když se rovná stringu „light” akorát nenastavuje parametry „theme” modelu „User_attachments” (viz. kapitola 3.4.1) na string „white”, ale na string „black”. Nakonec metoda vrátí metodu „redirect” s parametrem „/”. (Code With Tomi, 2021)

Další z metod je metoda „useable”, která bere jako parametr request. Tato metoda nejprve načte do pole „offer” všechny modely „Offer” (viz. kapitola 3.4.1), které mají parametr „expired” nastavený na False. Dále metoda kontroluje, zda pole „POST”, které je obsaženo v requestu obsahuje část pod jménem „category”. Pokud ano načte její hodnotu a uloží ji do proměnné „category_id”. Program zkontroluje, zda není proměnná „category_id” rovna stringu „none” a pokud není načte do proměnné „c” model „Category” (viz. kapitola 3.4.1) u kterého se parametr „id” rovná proměnné „category_id”. Dále už jen metoda projde celým polem „offer” a vyhodí z něj všechny modely „Offer” (viz. kapitola 3.4.1) u kterých parametr „category” není roven výše načtené proměnné „c”. V další části programu je

zkontrolováno, zda pole „POST" obsaženo v requestu obsahuje část pod jménem „keyword" a pokud ano je tato hodnota načtena do proměnné „keyword". Dále je zkontrolováno, zda proměnná není prázdná a pokud není z pole „offer" jsou vyhozeny všechny modely „Offer" (viz. kapitola 3.4.1) jejichž parametr „Title" neobsahuje proměnnou „keyword". (Ivy, 2021) V další části kódu je zkontrolováno hned několikrát zda pole „POST" obsahuje část pod jménem „any-price", „0-1000-price", „1001-2500-price", „2501-5000-price", „5001-10000-price", „10000-more-price", nebo zda neobsahuje „min-price" a zároveň „max-price". Pokud obsahuje hodnotu pod jménem některých ze stringů ve formátu „číslo-číslo-price" je z pole „offers" vyhozen každý model „Offer" (viz. kapitola 3.4.1), jehož parametr „price" neobsahuje hodnotu integeru mezi těmito dvěma čísly. Pokud je pole však obsahuje pole hodnotu pod jménem „max-price" a „min-price" jsou tyto dvě hodnoty načteny do proměných „max" a „min" respektivě. Dále je zkontrolováno, zda tyto hodnoty nejsou prázdné a pokud nejsou jsou z pole „offers" vyhozeny všechny takové modely „Offer" (viz. kapitola 3.4.1) jejichž parametr „price" není menší než proměnná „max", nebo větší než proměnná „min". Následuje poslední podmínka, která zjišťuje, zda obsahuje pole „POST" hodnotu pod jménem „distance" a pokud ano načte ji do proměnné „distance". Dále je zkontrolováno, zda proměnná není prázdná a pokud není je proiterováno pole „offer" a pro každou jeho položku je zkontrolováno, zda metoda range s parametry proměnná „distance", model „User" aktuálního uživatele a model „o", který je jednou z položek pole „offer". Pokud metoda vrátí False je iterovaná položka vyřazena z pole „offer". Předposlední částí je protiterování pole „offer" a zavolání metody „is_expired" s každou jeho položkou. Pokud metoda vrátí True je položka z pole vyjmuta. Poslední částí metody je podmínka, která zkontroluje, zda pole není prázdné. Pokud není vrátí samotné pole, ale pokud je vrátí string „failed".

Metoda „is_expired" je další krátkou metodou, která bere jako parametr model „Offer" (viz. kapitola 3.4.1). Nejprve metoda zkontroluje, zda neobsahuje parametr „creation_date" datum starší než 30 dní a pokud obsahuje nastaví parametr „expired" modelu na True, změnu uloží a vrátí True. Pokud však datum není starší pouze vrátí False.

Další z metod obsažených v souboru je metoda „importance", která bere jako parametr pole „offers" naplněné modely „Offer" (viz. kapitola 3.4.1). Nejprve metoda zkontroluje, zda nedostala do proměnné „offers" namísto pole string o hodnotě „failed" a pokud ne proiteruje pole „offers". U každé položky zkontroluje, zda je její parametr „importance" větší než nula a pokud je přidá ho do nového pole „importance" a vyřadí ho z původního pole „offers". Po iterování pole následovně metoda za pomocí importované metody „sorted" srovná položky pole „importance" podle jejich parametru „importance" od nejvyšší k nejnižší hodnotě a uloží nové pole do proměnné „v". následovně metoda už jen vrátí pole „v" a upravené pole „offers" spojené do jednoho pole. Pokud však proměnné offers obsahovala string „failed" metoda pouze vrátí string „failed".

Metoda „userRating” je další z metod obsáhlých v tomto souboru. Tato metoda bere jako parametr proměnnou „order_id”, proměnnou „rating”, proměnnou „comment” a model „User” uložený v proměnné „creator”. Metoda nejprve načte do proměnné „o” model „Offer” (viz. kapitola 3.4.1), který má jako parametr „id” nastavenou proměnnou „order_id” a do proměnné subject načte parametr „creator” tohoto modelu. Program dále načte do proměnné „relations” model „Rating_Relation” (viz. kapitola 3.4.1), který má jako parametr „rating_subject” nastavenou proměnnou „subject” a jako „rating_creator” proměnnou „creator”. Dále je zkontrolováno, zda tento model existuje a pokud ano je vrácen string o hodnotě „failed”. Pokud však ne je dále zkontrolováno, zda existuje model „User_attachments” (viz. kapitola 3.4.1), který má jako parametr „user” nastavenou proměnnou „subject”. Pokud tento model existuje je načten a do jeho parametru „rating” je načtena nová hodnota a ta je vypočítána podle přičtení proměnné „rating” k staré hodnotě a následovné vydělení dvěmi a nakonec je model uložen. Pokud však model neexistuje je vytvořen nový, kde je do parametru „user” nastavena hodnota proměnné „subject” a do parametru „rating” hodnota proměnné „rating”. Nakonec je v metodě vytvořen nový model „Rating_Relation” (viz. kapitola 3.4.1) a jeho parametrům „comment”, „rating_subject” a „rating_creator” jsou nastaveny hodnoty proměnných „comment”, „subject” a „creator” jednotlivě a je vrácen string o hodnotě „success”.

Další metoda „range” bere jako parametry proměnou „distance”, model „User” v proměnné „user” a model „Offer” (viz. kapitola 3.4.1) v proměnné „order”. Nejprve metoda načte do proměnných „att1” a „att2” model „User_attachmetns” (viz. kapitola 3.4.1), do první proměnné načte model s parametrem „user” rovnému proměnné „user” (brané jako parametr pro metodu) a do druhé model s parametrem „user” rovnému parametru „creator” modelu „Offer” (viz. kapitola 3.4.1) uloženého v proměnné „order”. Dále metoda vytvoří dvě proměnné „place1” a „place2” a do obou nastaví to co vrací metoda „place_spliting” s parametrem „att1” a „att2”. Metoda poté vytvoří dvě proměnné „url1” a „url2” do kterých dá string o hodnotě „https://api.geoapify.com/v1/geocode/search?text=” spojený s proměnnou „place1”, nebo „place2” a spojený se stringem „&apiKey=” (poslední část stringu má za znakem rovná se speciální klíč). Metoda dále vytvoří python „dictionary” do proměnné „headers” a nestaví do něj hodnotu, kterou vrací importovaná metoda „CaseInsensitiveDict”. Poté pod klíč „Accept” string „application/json”. Program poté opět vytváří dvě proměnné a to sice „resp1” a „resp2”, které zavolají metodu importované třídy „requests” „get” s parametry „url1”, nebo „url2” a do proměnné metody „get” „headers” nastaví proměnnou „headers”. Následovně jsou vytvořeny další dvě proměnné „data1” a „data2” do, kterých jsou dány, to co vrací metody „json” proměnných „resp1” a „resp2”. Metoda následovně vytvoří čtyři proměnné „lat1”, „lon1”, „lat2” a „lon2”. Do proměnné „lat1” a „lat2” jsou dány hodnoty proměnný „data1” a „data2” pod klíčemy „features”, „0”, „properties” a „lat”. Do proměnných „lon1” a „lon2” jsou nahrány hodnoty z proměnných „data1” a „data2” pod stejným klíčemy s rozdílem posledního klíče, který je „lon”. Dále jsou vytvořeny dvě pole „coords_1” a „coords_2” první hodnotách „lat1” a „lon1” a druhé o hodnotách „lat2” a „lon2”. Do poslední vytvořené proměnné „v” je nahrána hodnota, kterou vrací metoda

„geodesic” třídy „geopy” s parametry „coords_1” a „coords_2”. Dále už je jen zkontrolováno, zda je proměnná „v” menší, nebo rovna než proměnná „distance” (braná jako parametr) a pokud podmínka projde je vrácena hodnota True a pokud ne False.

Poslední metoda „place_splitting” bere do proměnné „att” model „User_attachments” (viz. knihovna 3.4.1). Nejprve je vytvořena proměnná „string” s prázdným stringem. Poté je do pole „city” je přidaná hodnota parametru „City” modelu „User_attachments” (viz. knihovna 3.4.1), která je rozdělena pomocí metody „split”. Následovně je pole proiterováno a do stringu „string” přidaná hodnota jedné části pole „city” se stringem „%20” a po iterování je přidán do stringu „string” string „%2C%20”. Dále je tento proces opakován ještě dvakrát s parametry „Street” a „Postal_code” modelu „User_attachments” (viz. knihovna 3.4.1) branného jako parametr metodou. Nakonec je ke stringu „string” přidán string „Czech%20Republic” a string je vrácen.

3.5 profilepage

Aplikace profilepage se stará o část aplikace, kterou může uživatel používat pouze, když je přihlášen. Uživatel zde může upravovat své osobní informace a smazat svůj účet a vytvářet, upravovat a mazat nové inzeráty.

3.5.1 urls.py

Tři základní části pole „urlpatterns" jsou cesty „personal_info", „offers" a „edit_offer", které vedou k jednotlivým hlavní částem aplikace profilepage. Cesta „personal_info" je volána, když chce uživatel změnit své osobní údaje a volá metodu „personal_info" (viz. kapitola 3.5.3), cesta „offers", když chce uživatel přidat nový inzerát a volá metodu „offers" (viz. kapitola 3.5.3), a cesta „edit_offer", když uživatel chce upravit inzerát a volá metodu „offer" (viz. kapitola 3.5.3).

```
urlpatterns = [
    path("", views.personal_info, name="personal_info"),
    path("offers/", views.offers, name="offers"),
    path("edit_offer/<int:offer_id>/", views.offer, name="edit_offer"),
]
```

obr. 26: Základní cesty aplikace profilepage

Další tři cesty jsou zde, aby byla umožněna správná funkčnost nakupování zvýraznění jednotlivých inzerátů. Tyto cesty jsou „confirmed", „canceled" a „paypal_redirect", cesta „confirmed" se volá, pokud platba prošla a je volána metoda „confirmed" (viz. kapitola 3.5.3), cesta „canceled" je volána, pokud platba neprošla a volá metodu „cancel" (viz. kapitola 3.5.3) a „paypal_redirect" zobrazuje mezistránku, kde uživatel potvrzuje svou platbu a volá metodu „payment_redirect" (viz. kapitola 3.5.3). (THE PROTON GUY, 2023)

```
urlpatterns = [
    path("confirmed/<str:payment_id>", views.confirmed, name="confirmed"),
    path("canceled/<str:payment_id>/", views.cancel, name="canceled"),
    path(
        "payment_redirect/<int:offer_id>",
        views.payment_redirect,
        name="paypal_redirect",
    ),
]
```

obr. 27: Cesty používané pro funkčnost nakupování

Posledními částmi pole jsou tři cesty „delete”, „delete_offer” a „refresh”. Jsou to cesty, které pouze provádí některou z jednoduchých akcí. Cesta „delete” se volá, když uživatel potřebuje smazat svůj účet a je volána metoda „delete” (viz. kapitola 3.5.3), „delete_offer”, když uživatel potřebuje smazat inzerát a je volána metoda „delete_offer” (viz. kapitola 3.5.3) a „refresh” používaná, pokud uživatel potřebuje obnovit inzerát a volá metodu „refresh” (viz. kapitola 3.5.3).

```
urlpatterns = [
    path("delete/", views.delete, name="delete"),
    path("delete/<int:offer_id>/", views.delete_offer, name="delete_offer"),
    path("refresh/<int:offer_id>", views.refresh, name="refresh"),
]
```

obr. 28: Cesty používané pro jednoduché akce

3.5.2 forms.py

Prvním formulářem, který soubor obsahuje, je formulář „MakeAnOffer” se čtyřmi parametry. První parametr je „title” (CharField), který má nastavený „label” na hodnotu „Title” a maximální délku na 255 znaků, druhým „category” (ModelChoiceField), který má nastavené vybíratelné objekty na všechny objekty modelu „Category” (viz. kapitola 3.4.1), a třetím je „price” (IntegerField). Posledním parametrem je „description” (RichTextUploadingFormField), který je nainportovaný z knihovny ckeditor_uploader.fields. (Smudja, 2020)

```
class MakeAnOffer(forms.Form):
    title = forms.CharField(label="Title", max_length=255)
    description = RichTextUploadingFormField()
    category = forms.ModelChoiceField(Category.objects.all())
    price = forms.IntegerField()
```

obr. 29: Formulář „MakeAnOffer”

Druhým formulářem je „adress”, který obsahuje tři parametry. První z parametrů je „City” (CharField), druhý je „Street” (CharField) a třetí „Postal_code” (CharField). Všechny tyto parametry mají nastavenou maximální délku na 255 znaků.

```
class adress(forms.Form):
    City = forms.CharField(max_length=225)
    Street = forms.CharField(max_length=225)
    Postal_code = forms.CharField(max_length=225)
```

obr. 30: Formulář „adress”

Poslední z formulářů je „edit_description”. Tento formulář má pouze jeden parametr a to sice „description” (RichTextUploadingFormField), který je také importovaný z třídy ckeditor_uploader.fields. (Smudja, 2020)

3.5.3 views.py

První metodou, kterou obsahuje soubor „views.py” je „personal_info”. Metoda má před sebou dekorátor „@login_required”, který omezuje nepřihlášeného uživatele od přístupu k této stránce. (Coderbook, 2019) První část metody je využívaná k měnění osobních informacích o uživateli. Nejprve je načten model „User_attachments” (viz. kapitola 3.4.1) do proměnné „att”. Program poté zkontroluje, zda request obsahuje metodu „POST” díky čemuž je možno potom načítat z requestu jednotlivé uživatelem zadané informace. Další částí kódu jsou podmínky, které zjišťují, zda uživatel zadal některé informace a pokud je zadal mění podle nich příslušné parametry modelu „User_attachments” (viz. kapitola 3.4.1). Tyto parametry jsou „phone_number”, „first_name”, „last_name”, „City”, „Street” a „Postal_code”. Další část metody, která je také zabalená v podmínce, která zjišťuje, zda request obsahuje metodu „POST” načítá vyplněný formulář „adress” (viz. kapitola 3.5.2) a pokud je přidává podle něj do modelu „User_attachments” (viz. kapitola 3.4.1) do parametrů „City”, „Street” a „Postal_code” příslušné informace a následně jej ukládá. Poslední částí metody je už jen vytvoření kontextu, který je následně předán do templatu. Do kontextu jsou přidány modely „chat” (viz. kapitola 3.4.1), které uživatel s někým vede, modely „Rating_Relation” (viz. kapitola 3.4.1) u kterých je uživatel subjektem hodnocení, model „User_attachments” (viz. kapitola 3.4.1), který přísluší danému uživateli, a formulář „adress“ (viz. kapitola 3.5.2). Následuje už jen vrácení metody render s parametry request, stringem „profilepage/personal_info.html” a s proměnou „context”, která obsahuje vytvořený kontext.

Další metodou je metoda „offers” a stejně jako metoda „personal_info” má dekorátor „@login_required”. (Coderbook, 2019) V první části této metody program zkontroluje, zda existuje model „payment” (viz. kapitola 3.4.1), který má parametr „user” rovný aktuálnímu uživateli a parametr „completed” rovný False a pokud takový existuje odstraní jej. V další části program načte do proměnné „offers” všechny modely „Offer” (viz. kapitola 3.4.1), které mají parametr „creator” roven aktuálnímu uživateli a do proměnné „att” načte model „User_attachments” (viz. kapitola 3.4.1), který uživateli přísluší. Další část programu se zabývá spracování formuláře „MakeAnOffer” (viz. kapitola 3.4.1). Nejprve program zkontroluje, zda request obsahuje metodu „POST” a poté zda parametr v předchozí části kódu načteného modelu „offer_count” je menší nebo roven jedné. Dále program načte do proměnné „u” vyplněný formulář „MakeAnOffer” (viz. kapitola 3.4.1) z requestu. Program dále zkontroluje, zda je formulář validní a když ano vytvoří za jeho pomoci nový model „Offer” (viz. kapitola 3.4.1). Modelu jsou do parametrů „Title”, „price”, „description” a „category” přiřazeny hodnoty z, uživatelem vyplněného formuláře. Do dalších parametrů jsou přiřazeny vždy stejné hodnoty. Do parametru „creation_date” aktuální čas, „expired” hodnota False, „creator” aktuální uživatel, „importance” hodnota 0. Poslední parametr „preview” má přiřazenou hodnotu kterou vrátí metoda „getImage”, které je dán parametr z vyplněného formuláře „description”. Následovně je model uložen a k parametru „offer_count” modelu

„User_attachments” (viz. kapitola 3.4.1), který byl načten dříve v programu do proměnné „att”, je přičteno 1. Následuje už jen vytvoření kontextu, který je poté předán do templatu. První část kontextu je formulář „MakeAnOffer” (viz. kapitola 3.5.2), další proměnná „att”, předposlední proměnná „offers” a poslední proměnná „show_form”. Proměnná „show_form” je vytvořena před kontextem. Do proměnné je nastavena hodnota „True”, ale pokud jeden z parametrů „City”, „Street”, „Postal_code”, nebo „phone_number” obsahuje prázdný string model „User_attachments” (viz. kapitola 3.4.1), načtený předtím do proměnné „att”, je mu přiřazena hodnota „False”. Následuje už jen vrácení metody „render” s parametry request, stringem „profilepage/user_offers.html” a s proměnou „context”, který obsahuje vytvořený kontext.

Třetí metoda offer, která má stejně jako předchozí metody dekorátor „@login_required”. (Coderbook, 2019) V první části programu je nastavení proměnné „expired”. Nejprve je proměnné nastavena hodnota False, ale pokud projde podmínka, která kontroluje, zda existuje model „Offer” (viz. kapitola 3.4.1), který má parametr „id” stejný jako proměnná „offer_id” (proměnná „offer_id” je metodě dána jako parametr) a parametr „expired” nastaven na „True”, je jí nastavena hodnota „True”. Dále je načten model „Offer” (viz. kapitola 3.4.1) do proměnné „offer”. Dále program kontroluje, zda request obsahuje metodu „POST” a pokud ano kontroluje, zda request obsahuje data z, uživateli vyplněných „inputů”. Pokud některá z těchto podmínek projde nastaví program parametru „Title”, „price”, nebo „category” modelu „Offer” předem načteného uživatelův „input”. Další část programu opět kontroluje, zda request obsahuje metodu „POST”, pokud ano načte z requestu do proměnné „u” formulář „edit_description” (viz. kapitola 3.5.2). Program dále zkontroluje, zda je načtený formulář validní a pokud ano uloží jediný parametr formuláře do parametru „description” modelu „Offer” (viz. kapitola 3.4.1) načteného dříve v metodě do proměnné „offer” a uloží model. Následuje poslední část metody, která vytváří kontext. Kontext obsahuje proměnou „offer” načtenou dříve v metodě, proměnou „offer_id”, proměnou „expired” opět vytvořenou dříve v metodě, proměnou „category” obsahující všechny modely „Category” (viz. kapitola 3.4.1), formulář „edit_description” (viz. kapitola 3.5.2) a proměnou „att” obsahující model „User_attachments” (viz. kapitola 3.4.1) příslušící aktuálnímu uživateli. Následuje už jen vrácení metody render s parametry request, stringem „profilepage/your_order_detail.html” a s proměnou „context”, který obsahuje vytvořený kontext.

Další dvě významné metody jsou „confirmed” a „cancel”, které jako všechny předcházející metody mají dekorátor „@login_required”. (Coderbook, 2019) Metoda „cancel” pouze vrátí metodu „render” s parametry request a stringem „profilepage/payment_canceled.html”. Metoda „confirmed” je narozdíl od metody „cancel” komplikovanější. Nejprve metoda načte do proměnné „p” model „payment” (viz. kapitola 3.4.1), který má parametr „cid” rovný proměnné „payment_id” (tuto proměnnou bere metoda jako parametr). Program dále do proměnné „o” přiřadí hodnotu parametru „order” modelu „payment” (viz. kapitola 3.4.1) načteného do proměnné „p”. Metoda v podmínce poté zkontroluje, zda existuje model „payment” (viz. kapitola 3.4.1) s parametrem „cid” rovnému proměnné „payment_id” a pokud ano nastaví metoda parametr „completed” modelu

„payment” (viz. kapitola 3.4.1) načteného do proměnné „p” na True a k parametru „importance” modelu „Offer” (viz. kapitola 3.4.1) načteného do proměnné „o” přičte 3. Dále už jen metoda vrátí metodu „render” s parametry request a „profilepage/payment_confirmation.html”.

Následuje metoda „payment_redirect”, která má dekorátor „@login_redirect”. (Coderbook, 2019) Metoda nejprve použije knihovnu „uuid”, aby do proměnné „id” mohl nastavit univerzální kód pomocí metody „uuid4” třídy „uuid”. Následovně metoda vytvoří model „payment” (viz. kapitola 3.4.1) a do parametrů „user” nastaví model aktuálního uživatele, „order” model „Offer” (viz. kapitola 3.4.1), jehož parametr „id” je roven proměnné „offer_id” („offer_id” je proměnná, kterou bere metoda jako parametr) a „cid” nastaví na vytvořenou proměnou „id”. Dále program vytvoří pythonový objekt „dictionary” s názvem „paypal_checkout”. Tento objekt obsahuje klíč a obsah, kde obsah je možno nalézt podle klíče. První dvojicí tohoto objektu je klíč „business” a obsah je proměnná „PAYPAL_RECEIVER_EMAIL”, která je proměnou importovaného souboru settings (viz. kapitola 3.3.1). Další dvojice jsou klíč „amount” a k němu přiřazena hodnota „10”, klíč „item_name” a hodnota „zvýraznění inzerátu”, klíč „currency_code” a hodnota „USD”, klíč „invoice” a hodnota nastavena pomocí metody „uuid4” knihovny uuid. Následují poslední tři dvojice první z nich je klíč „vnotify_url” a hodnota je string f”https”//{host}{reverse(‘paypal-ipn’)}”. Část stringu „{host}” do stringu přidá hodnotu proměnné „host”, která je předem nastavená na hodnotu, kterou vrací metoda objektu request „get_host()”. Další část „reverse(‘paypal-ipn’)}” vrátí string cesty pojmenované „paypal-ipn”. Druhá dvojice je klíč „return_url” a hodnota je string f”https”//{host}{reverse(‘confirmed’, kwargs = {‘payment_id’: id})}”. Tento string je stejný jako u předchozí dvojice s rozdílem toho, že metoda „reverse” tentokrát vrací cestu, která obsahuje parametr pro metodu, která je cestou volána. Poslední části je dvojice klíč „cancel_url” a hodnota string f”https”//{host}{reverse(‘canceled’, kwargs = {‘payment_id’: id})}”. Tato hodnota je stejná s rozdílem toho, že metoda „reverse” vrací cestu k jiné metodě. Poslední částí celé metody je už pouhé vytvoření kontextu. Kontext obsahuje model „User_attachment” příslušící aktuálnímu uživateli. Druhou a poslední částí kontextu je formulář „PayPalPaymentsForm” importovaný z knihovny „django-paypal”, který má při vytváření nastavený parametr “initial” na objekt „dictionary” označený jako „paypal_checkout”. Následovně už pouze metoda vrátí metodu render s parametry request, stringem „profilepage/payment_confirmation.html” a s vytvořeným kontextem. (THE PROTON GUY, 2023)

V souboru následují dvě metody „delete” a „delete_offer”. První z metod, a to sice metoda „delete” nejprve zkontrolujte, zda je uživatel přihlášen a pokud ano nejprve smaže jeho účtu příslušející model „User_attachments” (viz. kapitola 3.4.1) a poté model „User” příslušící uživateli. Následovně metoda zavolá metodu „logout” z, parametrem request a následovně vrátí metodu redirect s parametrem „/” (string). Druhá metoda „delete_offer” nejprve podle proměnné „offer_id”, kterou bere jako parametr, získá model „Offer” (viz. kapitola 3.4.1) a smaže ho. Následovně program získá model „User_attachments” (viz. kapitola 3.4.1) aktuálního uživatele a od jeho parametru „offer_count” odečte číslo 1. Následovně metoda zjistí, zda existuje model „chat” (viz. kapitola 3.4.1) s parametrem

„offer_id” rovnému proměnné „offer_id” a pokud existuje tak ho smaže. Nakonec metoda vrátí metodu redirect s parametrem „/profilepage/offers/” (string).

Předposlední metodou je metoda „refresh”. Tato metoda nejprve do proměnné „offer” načte model „Offer” (viz. kapitola 3.4.1) s parametrem „id” rovným proměnné „offer_id”, kterou metoda bere jako parametr. Tomuto modelu následovně nastaví parametr „creation_date” na aktuální čas a parametr „expired” na hodnotu False. Tento model následně uloží a vrátí metodu redirect s parametrem „/profilepage/offers/” (string).

Poslední metoda je metoda „getImage”. Tato metoda nejprve vytvoří instanci třídy „BeautifulSoup”, která je naimportovaná z knihovny, s parametry „html” („html” je proměnná, kterou metoda bere jako parametr) a „html.parser” (string). Metoda dále do proměnné „img” vloží to co vrací metoda „find” třídy vytvořené na předchozím řádku s parametrem „img” (string). Metoda poté už jen zkontroluje, zda je proměnná „img” prázdná a pokud ano vrátí adresu „https://www.thermaxglobal.com/wp-content/uploads/2020/05/image-not-found.jpg" a pokud ne vrátí hodnotu proměnné „img”, která je pythonový objekt „dictionary”, podle klíče „img”.

3.6 Chat

Aplikace chat je nejjednodušší ze všech aplikací, které jsou součástí projektu. Je zde pouze pro řešení spojení uživatelů a tím pádem i lepší orientování v kódu.

3.6.2 urls.py

Pole urlpatterns v tomto souboru obsahuje pouze jednu cestu. Touto cestou je cesta vedoucí do jediné metody souboru „views.py" (viz. kapitola 3.6.3). Metoda, která je volána je metoda „Chat" (viz. kapitola 3.6.3). Tato cesta je pouze číslo inzerátu, ohledně kterého si dva uživatele píšou. Jako jméno má cesta nastavené „chat".

```
urlpatterns = [
    path("<int:offer_id>/<int:user_id>/", views.Chat, name="chat"),
]
```

obr. 31: Jediná cesta aplikace Chat

3.6.3 views.py

Jediná metoda tohoto souboru je metoda „Chat". Metoda Chat nejprve získá inzerát, podle id inzerátu a id jednoho z uživatelů účastnících se rozhovoru, které bere jako argumenty z cesty. Metoda poté pomocí jednoduché podmínky zjistí, jestli existuje komunikace ohledně tohoto inzerátu mezi těmito dvěma uživateli a pokud ano načte ho. Pokud neexistuje vytvoří takovýto nový rozhovor. Do nového vytvořeného modelu dosadí jako parametr „user_1" model aktuálního uživatele, „user_2" model uživatele, který nabídku vytvořil a do posledního parametru "offer_id" id, které metoda získala z cesty.

Další část metody se stará o posílání zpráv. Vezme z pole request hodnotu zprávy a vytvoří model „message" (viz. kapitola 3.4.1) do parametru „chat" (viz. kapitola 3.4.1) přiřadí hodnotu objektu rozhovoru načteného z předchozí části kódu, do parametru „message_sender" objekt uživatele, který správu napsal a do parametru „message" samotnou hodnotu správy.

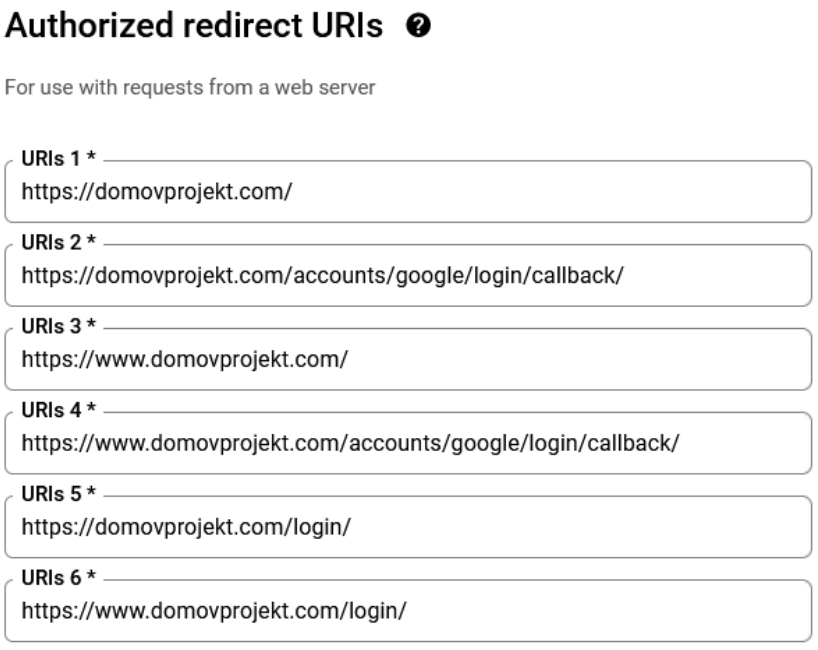
Poslední částí metody je vytvoření kontextu pro html souboru později vytvořeného pro uživatele. Do kontextu je přidán objekt rozhovoru mezi uživateli a modely všech zpráv. Následuje už jen vrácení metody „render" s templatem „chat.html".

4 Použité Api

4.1 Google OAuth

Google OAuth je služba využívaná v naší aplikaci pro zjednodušení přihlašování uživatelů. Musel být vytvořen speciální účet u googlu pro naší aplikaci. Nejprve musel být vytvořen ve webové aplikaci google cloud console nový projekt se jménem „django-google-login". V této aplikaci musel nejprve být vytvořen takzvaný OAuth consent screen se jménem „Django Login" a musel mu být nastaven email pro případné poskytnutí pomoci a dále nastavena doména, která je autorizovaná googlem. Tato doména je v našem případě „domovprojekt.com". Nakonec museli být vybrány informace, které budou googlem aplikaci poskytnuty (v našem případě email a jméno). (Akamai DevRel, 2022)

Dále muselo být vytvořen googlem nazvaný objekt „OAuth 2.0 Client IDs". Tento objekt se jmenuje „Bazaroos" a museli mu být nastaveny autorizované přesmněrovací domény. Tyto domény jsou nastaveny na hodnoty níže. Tento objekt následovně vygeneruje speciální dva parametry „client_id" a „secret_key", které jsou dále umístěny v souboru „config.json" (viz. kapitola 3.3.1). (Akamai DevRel, 2022)



obr. 32: Přesměrovací domány dané googlu

4.2 Paypal Api

Toto api je používáno pro možnost uživatelských plateb. Musel být vytvořen paypal účet pro umožnění této funkčnosti. Musel být zařízen nový sandbox „merchant", který je účet prodávajícího, účet a jeho údaje museli být nastaveny do souboru „config.json" (viz. kapitola 3.3.1) a ještě jeden účet, který musel být zařízen pro nakupujícího. Tyto dva účty jsou zařízeny pouze pro testování aplikace.

4.3 Geoapify Api

Toto api je využíváno v metodách souboru „views.py" (viz. kapitola 3.4.4) aplikace homepage v metodě „range". Api je využíváno k získávání souřadnic podle adresy. Finálním důvodem, proč je api používáno je potřeba počítat vzdálenost mezi jednotlivými uživateli a inzeráty. Pro účely této aplikace samozřejmě musel být vytvořen nový účet s emailem aplikace.

5 Nasazení projektu

Projekt je nasazen za pomocí služby azure od společnosti microsoft. Tento projekt konkrétně běží na virtuálním počítači o velikosti ve webové aplikaci nazývané jako „Standard B1s". Toto je pouze nevýkonný počítač o výkon pouze 1 jádra, objemu paměti RAM 1 gigabajtu a na počítači běží operační systém linux (konkrétně Ubuntu 20.04 Lts).

Na tomto počítači je nainstalován python, postgresql, nginx, gunicorn a supervisorctl. Python je zde používán pro správnou funkčnost samotné aplikace. To znamená virtuálního prostředí a kódu. Postgresql je zde jako databáze pro tento projekt. Gunicorn je nainstalovaný ve virtuálním prostředí a jeho správná funkčnost je zajišťována právě softwarem známým jako supervisorctl. Poslední částí projektu je nginx, který se stará o správné zacházení se statickými soubory jako jsou soubory typu „.css", „.js" a různé druhy obrázků. (Zahoor, 2023)

Pro správnou funkčnost tohoto projektu, které musely být upravené speciální soubory obsahující konfiguraci. Tyto soubory byli napsány pro supervisorctl a pro nginx. V souboru pro supervisorctl se pouze zmiňuje konfigurace gunicronu. V konfiguračním souboru se zmiňuje jak zacházet se statickým soubory a kde jsou umístěny. Oba soubory jsou uvedeny níže. (Zahoor, 2023)

```
server{
        server_name 98.71.34.139;
        server_name domovprojekt.com;

        location / {
                include proxy_params;
                proxy_pass http://unix:/home/janse/2023-3e-Bazar/app.sock;
        }
        location /static/ {
                autoindex on;
                alias /home/janse/2023-3e-Bazar/static/;
        }
        location /media/ {
                autoindex on;
                alias /home/janse/2023-3e-Bazar/media/;
        }
}
```

obr. 33: Konfigurační soubor nginxu

6 ZÁVĚR

Po dokončení práce na našem projektu se nám podařilo udělat funkční responzivní, dobře vypadající webovou stránku. Vznikla webová aplikace, která dokáže našim návštěvníkům zpříjemnit prostředí pro nakupování.

Práce na tomto projektu nám přinesla nejen možnost aplikovat a prohloubit naše znalosti v oblasti webového vývoje, ale také nám umožnila zkoumat problematiku online bazarů a webů jako takových.

Naše týmová spolupráce a odhodlání nám umožnily vytvořit uživatelsky přívětivou stránku.

Přestože jsme narazili na několik problému při vytváření této stránky, byli jsme schopni je úspěšně překonat díky našim nově naučeným dovednostem.

Celkově jsme spokojeni s naší prací a na výsledek kterého jsme dosáhli. Stránku se budeme do budoucna snažit aktualizovat a budeme se snažit web vylepšovat tak, aby přinesl pro naše uživatele ještě lepší a přívětivější prostředí.

Použitá literatura

Akamai DevRel. (2022, December 12). *Set up Google Sign-In for Faster Django Login Experience feat. Tech with Tim*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=yO6PP0vEOMc

Codemy.com. (2021, 6 30). *Login With User Authentication - Django Wednesdays #21*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=CTrVDi3tt8o&t=1128s

Coderbook. (2019, January 16). *How to restrict access with Django Permissions*. Coderbook. Retrieved April 14, 2024, from https://coderbook.com/@marcus/how-to-restrict-access-with-django-permissions/

Code With Tomi. (2021, July 14). *How To Add Light/Dark Theme In Django Using Database*. YouTube. Retrieved April 15, 2024, from https://www.youtube.com/watch?v=gHgCr6ctfSU

Digital World. (2023, May 18). *Python Django - Password Reset*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=hnfwCFRKqh8

Ivy, D. (2021, March 17). *Create A Search Bar - Django Wednesdays #9*. YouTube. Retrieved April 15, 2024, from https://www.youtube.com/watch?v=AGtae4L5BbI

Ivy, D. (2021, May 24). *How to Add reCAPTCHA to a Django Site*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=8rWXdkUn3PM

KenBroTech. (2021, July 8). *Customize Password Reset Mail in Django | Django Blog Project - Extra*. YouTube. Retrieved April 23, 2024, from https://www.youtube.com/watch?v=0pa75ch0S4E

THE PROTON GUY. (2023, September 1). *Django Paypal Payment Integration in 30 Minutes*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=LJ7pzebXX6g

Schafer, C. (2018, December 13). *Python Django Tutorial: Deploying Your Application (Option #1) - Deploy to a Linux Server*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=Sa_kQheCnds

Smudja, M. (2020, February 24). *How To Upload Images in Django - CKEditor Tutorial*. YouTube. Retrieved April 14, 2024, from https://www.youtube.com/watch?v=W61PvbzQaMw

stackpython. (2020, August 6). *How to Start Django Project with a Database(PostgreSQL)*. Medium. Retrieved April 14, 2024, from https://stackpython.medium.com/how-to-start-django-project-with-a-database-postgresql-aaa1d74659d8

Uddin, G. (2022, December 20). *How To Add A Custom Rich Text-Editor In Your Django Website*. Medium. Retrieved April 14, 2024, from

https://medium.com/@guddin93/how-to-add-a-custom-rich-text-editor-in-your-django-website-13914f048cc6

Wazy. (2021, 10 14). *Ckeditor - user friendly image upload*. stackoverflow. Retrieved April 14, 2024, from https://stackoverflow.com/questions/69498691/ckeditor-user-friendly-image-upload

Zahoor, H. (2023, March 12). *How to deploy Django on ubuntu with nginx and gunicorn*. Medium. Retrieved April 15, 2024, from https://medium.com/@huzaifazahoor654/how-to-deploy-django-on-ubuntu-with-nginx-and-gunicorn-9288b2c4e922

SEZNAM OBRÁZKŮ

1. Příklad třídy v souboru „models.py”
2. Načtení souboru „config.json” a jeho použití
3. Konfigurace cest k statickým a media souborům
4. Konfigurace cest k přihlášení
5. Pole „INSTALLED_APPS” obsahující nainstalované aplikace
6. Konfigurace potřebná k správné funkčnosti přihlašování
7. Konfigurace databáze
8. Konfigurace aplikace ckeditor
9. Konfigurace aplikace paypal
10. Konfigurace potřebná pro posílání emailů
11. Cesty k aplikacím
12. Cesty pro různé aplikace
13. Model Kategorije
14. Model inzerátu
15. Model přídavných informací k uživateli
16. Model „Rating_Relation”
17. Model diskuze dvou uživatelů
18. Model zprávy
19. Model platby
20. Základní cesty aplikace homepage
21. Cesty potřebné pro správnou funkčnost měnění hesel
22. Ostatní cesty souboru
23. Formulář pro přihlášení
24. Formulář pro zaregistrování
25. Formulář pro hodnocení
26. Základní cesty aplikace profilepage
27. Cesty používané pro funkčnost nakupování
28. Cesty používané pro jednoduché akce
29. Formulář „MakeAnOffer”
30. Formulář „adress”
31. Jediná cesta aplikace Chat
32. Přesměrovací domány dané googlu
33. Konfigurační soubor nginxu