

# **ZÁVĚREČNÁ STUDIJNÍ PRÁCE**

## **dokumentace**

### **Pokročilá administrace v Django**



**Autor:** Petr Mičola  
**Obor:** 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  
se zaměřením na počítačové sítě a programování  
**Třída:** IT4  
**Školní rok:** 2023/24



## ***Poděkování***

*Na úvod bych chtěl poděkovat panu učiteli Mgr. Marku Lučnému za ochotu a podporu při tvorbě tohoto projektu.*

## **Prohlášení**

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

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

V Opavě 1. 1. 2024

.....

Podpis autora



## Abstrakt

Tato práce se zaměřuje na vytvoření webové aplikace s autentizací uživatelů pomocí webového frameworku Django. Hlavním cílem projektu je implementace autentizace uživatelů prostřednictvím účtů GitHub a Microsoft, spolu s možností úpravy uživatelského profilu. Pro dosažení znovupoužitelnosti byl vyvinut backend, který může být integrován do jiných aplikací. Využití technologií jako Django, django-allauth, PostgreSQL a Docker zajišťuje robustnost a efektivitu aplikace. Gunicorn a Nginx byly zvoleny k obsluze webových požadavků, což přispívá k vysoké úrovni výkonu a škálovatelnosti. Soft UI Dashboard poskytuje moderní a přehledné prostředí administrace. Celková funkčnost aplikace je demonstrována na jednoduchém příkladu užití. Díky těmto implementacím má práce potenciál sloužit jako výchozí rámec pro vytváření dalších webových aplikací s podobnými požadavky na autentizaci uživatelů.

## Klíčová slova

Django, webová aplikace, autentizace uživatelů, úprava profilu, znovupoužitelný backend

## Abstract

This work focuses on creating a web application with user authentication using the Django web framework. The main project objective is to implement user authentication through GitHub and Microsoft accounts, along with the capability to edit user profiles. To achieve reusability, a backend has been developed, which can be integrated into other applications. Utilizing technologies such as Django, django-allauth, PostgreSQL, and Docker ensures the robustness and efficiency of the application. Gunicorn and Nginx have been chosen to handle web requests, contributing to a high level of performance and scalability. The Soft UI Dashboard provides a modern and clear administrative interface. The overall functionality of the application is demonstrated through a simple use case. Thanks to these implementations, the work has the potential to serve as a foundational framework for developing additional web applications with similar user authentication requirements.

## Keywords

Django, web application, user authentication, profile editing, reusable backend

# Obsah

<b>Úvod</b>	<b>3</b>
<b>1 Webová aplikace</b>	<b>5</b>
1.1 Úvod . . . . .	5
1.2 Statické a dynamické webové stránky . . . . .	5
1.3 Webový server . . . . .	6
1.4 Aplikační server . . . . .	6
1.5 Databáze . . . . .	7
<b>2 Framework</b>	<b>9</b>
2.1 Úvod . . . . .	9
2.2 Django . . . . .	9
2.3 Model-Pohled-Šablona . . . . .	10
2.4 Struktura projektu . . . . .	12
<b>3 Využité technologie</b>	<b>13</b>
3.1 Django . . . . .	13
3.2 Django-allauth . . . . .	13
3.3 PostgreSQL . . . . .	13
3.4 Docker . . . . .	14
3.5 Gunicorn . . . . .	14
3.6 Nginx . . . . .	14
3.7 Soft UI Dashboard . . . . .	15
<b>4 Způsoby řešení, použité postupy</b>	<b>17</b>
4.1 Vytvoření projektu . . . . .	17
4.2 Autentizace Django-allauth . . . . .	17
4.3 Vlastní uživatelský model . . . . .	18
4.4 Citlivá data . . . . .	18
4.5 Dockerizace aplikace . . . . .	18
4.6 Administrace . . . . .	19
4.7 Databáze PostgreSQL . . . . .	20
4.8 Příprava aplikace pro produkční účely . . . . .	20
<b>5 Výsledky řešení, uživatelský manuál</b>	<b>21</b>
5.1 Funkce aplikace . . . . .	21
5.2 Splněné a nesplněné cíle . . . . .	21



# ÚVOD

Mým cílem bylo vytvořit aplikační backend, který se bude starat o autentizaci uživatelů. Hlavním úkolem bylo umožnit uživatelům přihlašování skrze účty GitHub a Microsoft.

## **Proč jsem si vybral toto téma?**

Každý student naší školy má účty GitHub a Microsoft. Proto jsem se rozhodl pro tento projekt – chtěl jsem vytvořit autentizační backend, jež můžou učitelé naší školy použít pro své vlastní aplikace.

## **Proč zrovna tyto technologie?**

S webovým frameworkem Django jsme ve škole pracovali, což mi poskytlo solidní základy. Chtěl jsem využít svých znalostí ze studia a zároveň se něco naučit o ostatních technologiích, které řeší problematiku autentizace.

## **Struktura dokumentace**

Tato dokumentace popisuje tvorbu projektu od samotné první myšlenky a použitých technologiích až po výslednou realizaci.





# 1 WEBOVÁ APLIKACE

## 1.1 ÚVOD

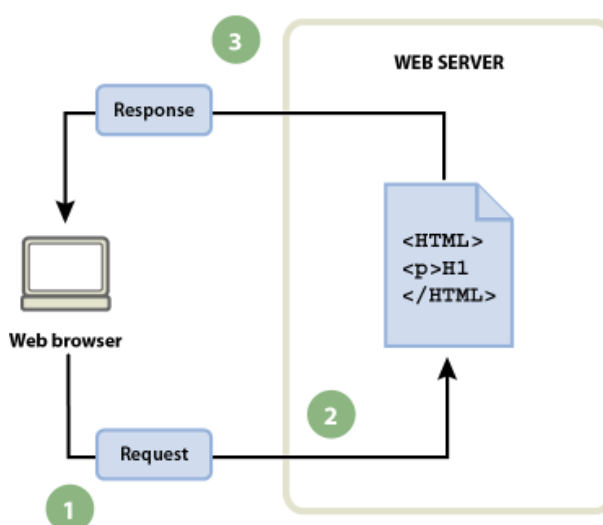
Webová aplikace je aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť Internet. Uživatelé mohou přistupovat k webovým aplikacím prostřednictvím webového prohlížeče, takže nemusí instalovat žádný speciální software na svých zařízeních.

## 1.2 STATICKÉ A DYNAMICKÉ WEBOVÉ STRÁNKY

Webová aplikace obsahuje stránky s částečně nebo úplně neurčeným obsahem. Konečný obsah stránky se určí až tehdy, když návštěvník požádá o stránku z webového serveru. Webová aplikace je kolekcí statických a dynamických webových stránek.

### 1.2.1 Zpracování statických stránek

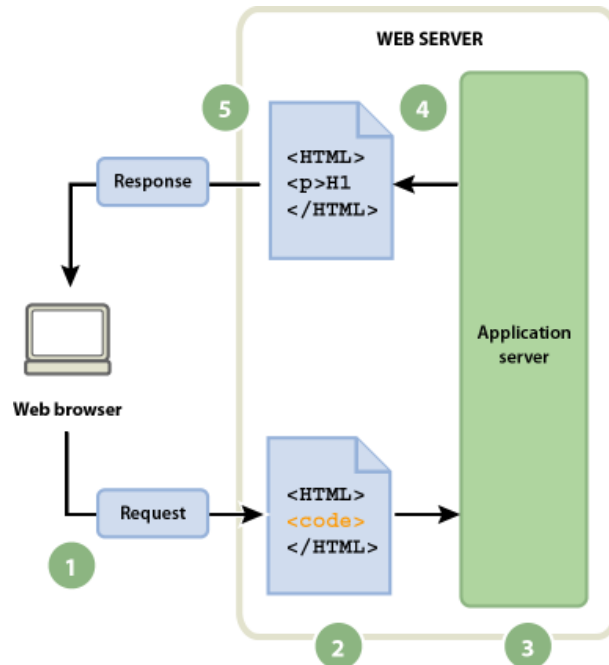
Když webový server přijme požadavek na statickou webovou stránku, pošle ji přímo prohlížeči, který o ni požádal. Statická webová stránka se nemění.



Obrázek 1.1: Zpracování statické webové stránky

### 1.2.2 Zpracování dynamických stránek

Naproti tomu, když webový server přijme požadavek na dynamickou stránku, předá stránku aplikačnímu serveru, který odpovídá za dokončení stránky. Aplikační server si přečte kód na stránce, dokončí stránku podle instrukcí v kódu a pak odstraní kód ze stránky.



Obrázek 1.2: Zpracování dynamické webové stránky

## 1.3 WEBOVÝ SERVER

Webový server je software, který posílá webové stránky na základě požadavků od webových prohlížečů. Požadavek na stránku se generuje, když návštěvník ve webovém prohlížeči klepne na odkaz na webové stránce, vybere záložku nebo zadá adresu URL do textového pole pro adresu.

## 1.4 APLIKAČNÍ SERVER

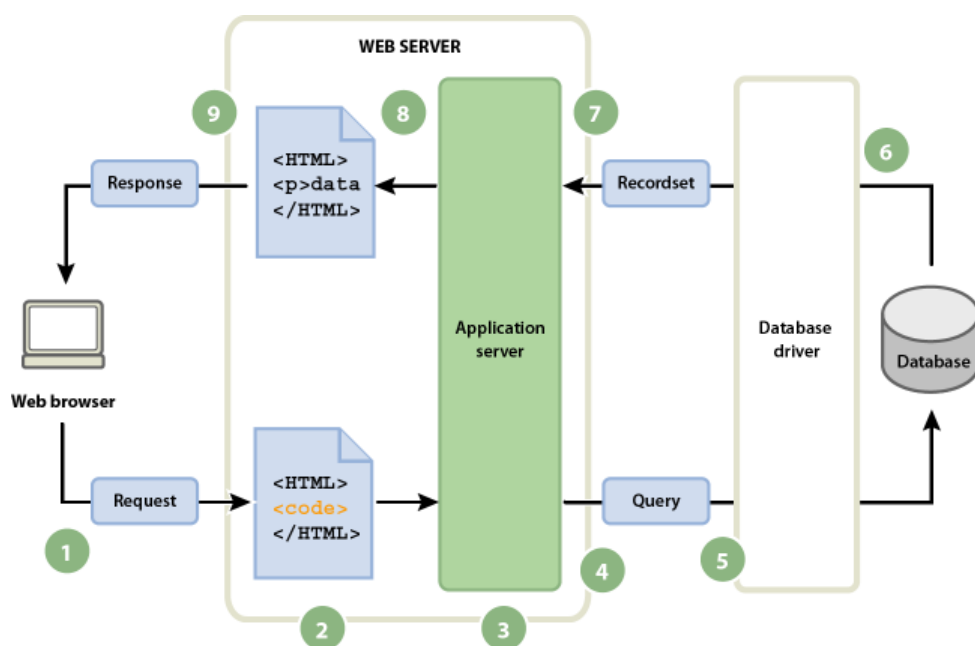
Aplikační server je software, který pomáhá webovému serveru zpracovat webové stránky obsahující skripty nebo tagy na straně serveru. Když webový server přijme požadavek na takovou stránku, předá stránku aplikačnímu serveru ke zpracování, než ji pošle prohlížeči.

## 1.5 DATABÁZE

Databáze je kolekce dat uložených v tabulkách. Každý řádek tabulky představuje jeden záznam a každý sloupec představuje pole v záznamu.

### 1.5.1 Přístupování k databázi

Aplikační server vám umožňuje pracovat s prostředky na straně serveru, jako jsou například databáze. Využitím databáze k uložení obsahu můžete návrh webu oddělit od datového obsahu, který chcete zobrazovat.



Obrázek 1.3: Přístupování k databázi



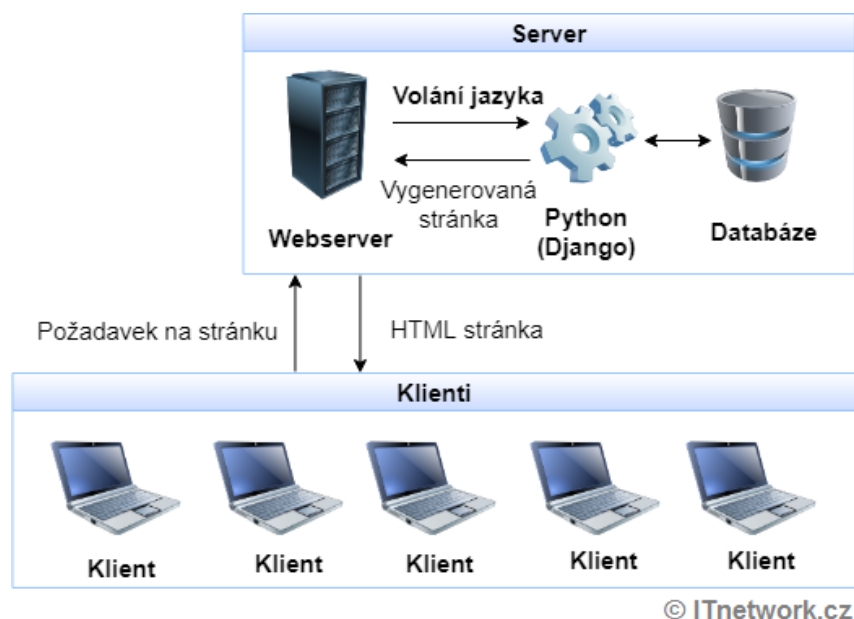
## 2 FRAMEWORK

### 2.1 ÚVOD

Framework je softwarová struktura pro podporu programování, vývoje a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji. Cílem frameworku je převzetí typických problémů dané oblasti, což umožní, aby se návrháři a vývojáři mohli soustředit pouze na své zadání.

### 2.2 DJANGO

Django IPA je open source webový aplikační framework napsaný v Pythonu, který se volně drží architektury Model-Pohled-Šablona. Hlavním úkolem Django je snadné vytvoření komplexních, databází řízených webových aplikací. Zaměřuje se na znovupoužitelnost a propojitelnost komponent, rychlý vývoj, v duchu „DRY“ (Don't Repeat Yourself) – neopakovat se.



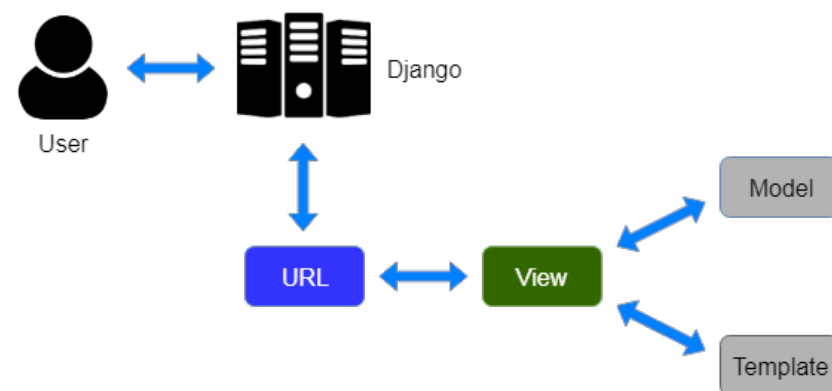
Obrázek 2.1: Diagram frameworku Django

## 2.2.1 Historie

Django bylo původně navrženo pro správu několika zpravodajsky orientovaných stránek společnosti The World Company v Lawrenci v Kansasu; později, v červnu 2005, bylo vydáno veřejně pod open-sourceovou licencí BSD. Framework byl pojmenován po jazzovém kytaristovi Django Reinhardtovi.

## 2.3 MODEL-POHLED-ŠABLONA

Model-View-Controller a je architektonický vzor, který je běžně používán při vývoji webových aplikací. V případě frameworku Django se tato architektura označuje jako Model-Pohled-Šablona (Model-View-Template).



Obrázek 2.2: Architektura Model-Pohled-Šablona

### 2.3.1 Model

Model reprezentuje datovou strukturu aplikace. Může to být například databázová tabulka, kde jsou ukládána data. Ve frameworku Django se modely definují jako třídy, které dědí od předdefinovaných modelových tříd poskytovaných frameworkem.

```
1 # models.py
2 from django.contrib.auth.models import AbstractUser
3 from django.db import models
4 class CustomUser(AbstractUser):
5     email = models.EmailField(unique=True)
6     username = models.CharField(unique=True, max_length=30)
7     profile_picture = models.ImageField(upload_to='profile_pictures/')
```

Kód 2.1: Příklad modelu Uživatel

### 2.3.2 Pohled

Pohled je část, která zpracovává požadavky od uživatelů a reaguje na ně. Obsahuje logiku pro získání dat z modelu a přípravu dat pro zobrazení. V Django se pohledy jsou implementovány jako funkce nebo třídy.

```
1  # views.py
2  from django.shortcuts import render
3  def quiz(request):
4      user = request.user
5      return render(request, 'quiz/quiz.html', {'user': user})
```

Kód 2.2: Příklad pohledu Kvíz

### 2.3.3 Šablona

Šablona definuje, jak jsou data zobrazena uživateli. Jedná se o prezentaci, která může obsahovat HTML, CSS a speciální značky nebo proměnné, které jsou nahrazeny konkrétními daty během zpracování. Ve frameworku Django jsou šablony soubory s příponou .html, které oddělují prezentaci od logiky pohledu.

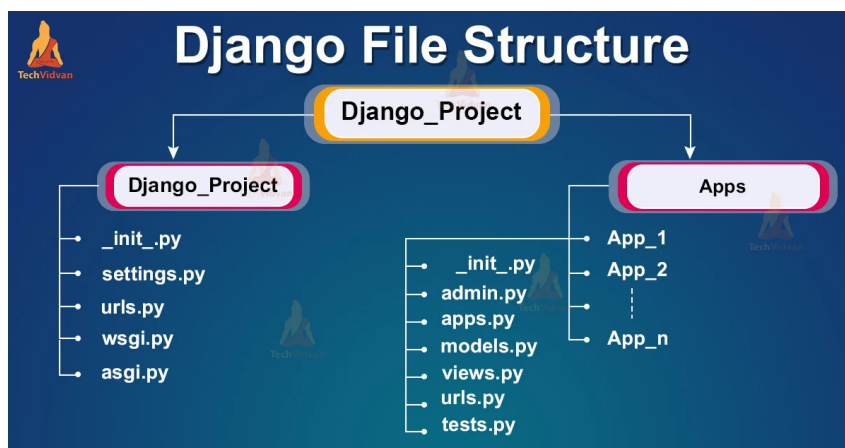
```
1  # base.html
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>DSA - {% block head_title %}{% endblock head_title %}</title>
8      <link href='{% static "style.css" %}' rel='stylesheet'>
9  </head>
10  {% include 'components/navbar.html' %}
11  <body>
12  {% block content %}
13  {% endblock content %}
14  </body>
15  {% include 'components/footer.html' %}
16  </html>
```

Kód 2.3: Příklad základní šablony



## 2.4 STRUKTURA PROJEKTU

Projekty v Django mají specifickou strukturu, která pomáhá organizovat kód, šablony, statické soubory a další komponenty. Po instalaci Django se nový projekt vytvoří příkazem `django-admin startproject (název projektu)`.



Obrázek 2.3: Struktura projektu Django

### 2.4.1 Soubory projektu

Adresář projektu po první konfiguraci bude obsahovat tyto soubory:

- `manage.py` – Script, který má na starosti správu projektu,
- `settings.py` – Konfigurační script společný všem aplikacím v projektu,
- `urls.py` – Globální konfigurace URL.

### 2.4.2 Soubory aplikace

Vytvoření struktury nové aplikace proběhne po spuštění příkazu `python manage.py startapp (název aplikace)`. Po spuštění tohoto scriptu se vytvoří podadresář se strukturou:

- `views.py` – Obsahuje jednotlivé view funkce,
- `urls.py` – Obsahuje mapování URL na jednotlivá view,
- `models.py` – Obsahuje popis datového modelu aplikace,
- `tests.py` – Obsahuje jednotkové testy.

## 3 VYUŽITÉ TECHNOLOGIE

### 3.1 DJANGO

Django IPA je open source webový aplikační framework napsaný v Pythonu, který se volně drží architektury Model-Pohled-Šablona.



Obrázek 3.1: Django

### 3.2 DJANGO-ALLAUTH

Django-allauth je integrovaná sada aplikací Django řešící autentizaci, registraci a správu účtů třetích stran.

### 3.3 POSTGRESQL

PostgreSQL je objektově-relační databázový systém. Na jeho vývoji se podílí globální komunita vývojářů a firem.



Obrázek 3.2: PostgreSQL

## 3.4 DOCKER

Docker je open source software, jehož cílem je poskytnout jednotné rozhraní pro izolaci aplikací do kontejnerů.



Obrázek 3.3: Docker

## 3.5 GUNICORN

Gunicorn je open-source WSGI server napsaný v Pythonu, používaný pro spouštění webových aplikací. Jeho hlavním cílem je poskytovat efektivní a spolehlivé zpracování HTTP požadavků.



Obrázek 3.4: Gunicorn

## 3.6 NGINX

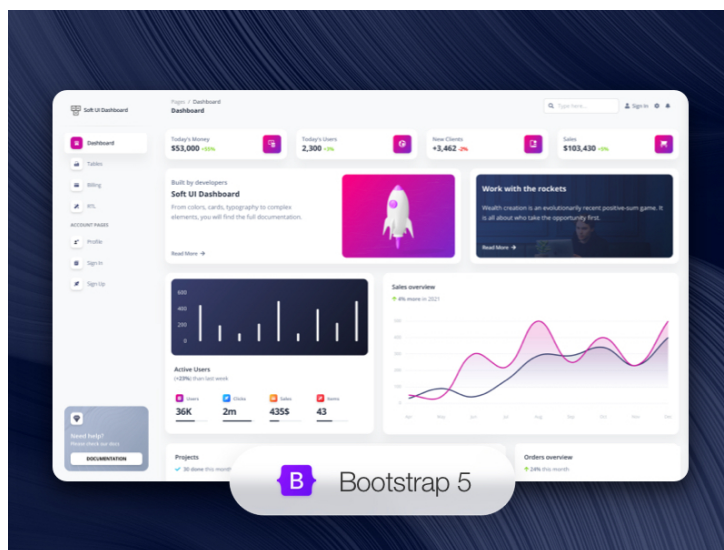
Nginx je softwarový open source webový server. Pracuje s protokoly HTTP (HTTPS), SMTP, POP3, IMAP a SSL. Zaměřuje se především na vysoký výkon a nízké nároky na paměť.



Obrázek 3.5: Nginx

## 3.7 SOFT UI DASHBOARD

Soft UI Dashboard je šablona administrace vytvořená pomocí Bootstrap 5.



Obrázek 3.6: Soft UI Dashboard



## 4 ZPŮSOBY ŘEŠENÍ, POUŽITÉ POSTUPY

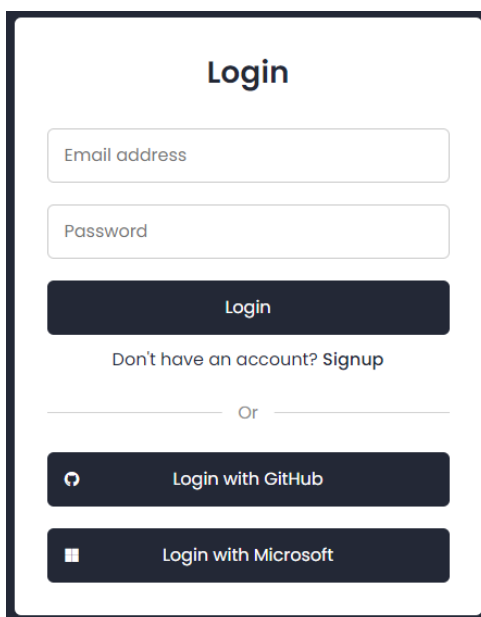
Při hledání vhodné technologie pro tuto aplikaci jsem narazil na Django. Tento webový aplikační framework je v problematice autentizace často využíván. S Djangem jsem navíc pracoval ve škole, tak jsem se rozhodl mé znalosti využít a Django použít.

### 4.1 VYTVOŘENÍ PROJEKTU

Prvním krokem bylo vytvoření projektu. Stačilo nainstalovat Python a framework Django. Příkazem se poté vytvořila základní struktura.

### 4.2 AUTENTIZACE DJANGO-ALLAUTH

Dále bylo mým úkolem umožnit uživatelům registraci a přihlášení pomocí aplikací třetích stran (GitHub a Microsoft). Pro tuto funkci jsem zvolil balíček django-allauth, který proces vytváření autentizace značně urychlil a autentizaci zabezpečil.



The image shows a login form with the following elements:

- Title:** Login
- Input Fields:** Email address, Password
- Buttons:** Login
- Text:** Don't have an account? Signup
- Separator:** Or
- Social Login Buttons:** Login with GitHub, Login with Microsoft

Obrázek 4.1: Formulář přihlášení

### 4.2.1 Aplikace třetích stran

Balíček django-allauth bylo potřeba nainstalovat. Poté jsem musel udělat pár změn v settings.py. Pro funkčnost přihlášení aplikací třetích stran jsem musel tyto aplikace přidat do struktury SOCIALACCOUNT\_PROVIDERS. K tomu jsem potřeboval klíče těchto poskytovatelů, ty jsem získal zaregistrováním a nastavením aplikací OAuth.

### 4.2.2 Kontrola funkčnosti

Pro kontrolu funkčnosti jsem django-allauth musel přidat do cest urls.py a vytvořit jednoduchý pohled ve views.py, který vracel šablonu základní stránky.

## 4.3 VLASTNÍ UŽIVATELSKÝ MODEL

Pro využití vlastního modelu uživatele jsem v models.py vytvořil model User, který vycházel z Django AbstractUser. Tomuto modelu jsem přiřadil email, uživatelské jméno a profilové foto, vše s patřičnými parametry. Model jsem nakonec musel zaregistrovat v admin.py aplikace a nastavit v settings.py.

## 4.4 CITLIVÁ DATA

Dalším úkolem bylo aplikaci zdockerizovat. Před tím jsem se však musel postarat o citlivá data (klíče aplikací třetích stran a Django samotného). Tyto data jsem se ze souborů rozhodl skrýt a všechna umístit do jednoho souboru .env.

### 4.4.1 Python-decouple

Abych poté mohl k datům přistupovat nainstaloval jsem balíček python-decouple. Následně stačilo z balíčku importovat object config a citlivé proměnné nahradit config("název proměnné v souboru .env").

## 4.5 DOCKERIZACE APLIKACE




Dockerizace je proces přizpůsobení a zapouzdření aplikace do kontejneru pomocí technologie Docker. Kontejner je samostatný a izolovaný balíček, který obsahuje veškeré potřebné soubory aplikace.

## 4.5.1 Docker Desktop

Jako první jsem si nainstaloval program Docker Desktop, který umožňuje práci s kontejnery v grafickém prostředí.

## 4.5.2 Docker Compose

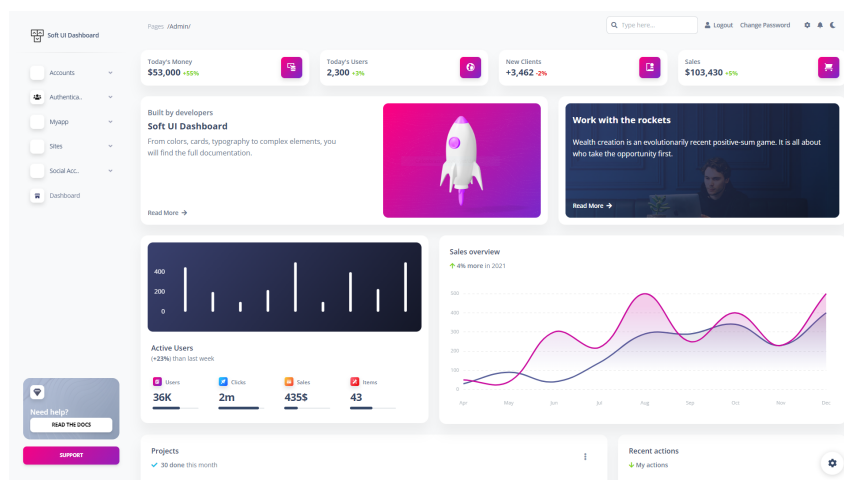
Dále jsem také využil nástroj Docker Compose, díky němuž jsem mohl definovat a spouštět více kontejnerů současně jako součást jedné aplikace. V mém případě to byl jeden kontejner s databází a jeden kontejner se samotnou aplikací.

Name	Image
 <a href="#">zaverecna-prace-main</a>	
 <a href="#">web-1</a> 6390180fef30	<a href="#">zaverecna-prace-main</a>
 <a href="#">db-1</a> 163b2e2a3373	<a href="#">postgres:15</a>

Obrázek 4.2: Docker kontejnery

## 4.6 ADMINISTRACE

Pro administraci uživatelů jsem se rozhodl použít původní administraci Django a upravit si její vzhled podle mých preferencí.



Obrázek 4.3: Administrace na adrese /admin



### **4.6.1 Šablona Soft UI Dashboard**

K úpravě administrace jsem vybral šablonu vytvořenou pomocí Bootstrap 5 Soft UI Dashboard. Tuto technologii lze získat dvěma způsoby - stáhnutím souborů z GitHubu nebo nainstalováním balíčku `django-admin-soft-dashboard`. Rozhodl jsem se využít druhou možnost a balíček nainstaloval.

## **4.7 DATABÁZE POSTGRESQL**

Django v základním nastavení používá databázi SQLite, ta ale pro produkční účely není vhodná. Databázi SQLite jsem proto vyměnil za velice populární databázi PostgreSQL, která má dobrou podporu Djangem. Databázi jsem vytvořil a propojil s aplikací.

### **4.7.1 Docker entrypoint**

Vytvoření souboru `entrypoint.sh`, který využívá Docker entrypoint přišlo vhod, protože tento soubor zkontroluje funkčnost databáze předtím, než se aplikace nainstaluje.

## **4.8 PŘÍPRAVA APLIKACE PRO PRODUKČNÍ ÚČELY**

Aby aplikace mohla v budoucnu efektivně fungovat také v produkci, musel jsem dodělat pár úprav.

### **4.8.1 Gunicorn**

Rozhodl jsem se využít Gunicorn, aby aplikace mohla efektivně a rychle zpracovávat webové požadavky.

### **4.8.2 Produkční Dockerfile**

Dále jsem upravil soubor s citlivými daty a vytvořil Dockerfile speciálně pro produkci.

### **4.8.3 Nginx**

Nginx použitý jako reverse proxy pro Gunicorn mi umožnil efektivní přijímání požadavků od klientů a řešení statických souborů.

## **5 VÝSLEDKY ŘEŠENÍ, UŽIVATELSKÝ MANUÁL**

Výsledkem práce je webová aplikace v Django, která řeší autentizaci a administraci uživatelů.

### **5.1 FUNKCE APLIKACE**

#### **5.1.1 Uživatel**

Když uživatel navštíví webovou stránku, vyskočí na něj okno s varováním, že není přihlášený. Uživatel se může přihlásit nebo zaregistrovat. K tomu může použít přihlášení aplikace třetích stran, GitHub nebo Microsoft. Po úspěšné autentizaci se uživateli zobrazí profilová stránka, kde uvidí svou profilovou fotku, uživatelské jméno a email. Uživatel se následně může odhlásit nebo upravit svůj profil. Když zůstane přihlášený a vrátí se na stránku kvízu, kvíz se mu zobrazí.

#### **5.1.2 Administrátor**

Administrátor aplikace může přejít na stránku /admin, kde se musí přihlásit. Do této administrace má přístup pouze administrátorský účet, prostý uživatel má přístup zamítnut. Když se administrátor přihlásí, může si zobrazit seznam uživatelů aplikace a upravit jejich profil manuálně přímo z administrace.

### **5.2 SPLNĚNÉ A NESPLNĚNÉ CÍLE**

Splněné cíle:

- Funkční autentizační webová aplikace v Django,
- umožnění uživatelům autentizaci skrze účty GitHub a Microsoft a úpravu jejich profilu,
- poskytnutí znovupoužitelného backendu pro jiné aplikace.

Nesplněné cíle:

- Běh aplikace na produkčním webovém serveru.



## 6 ZÁVĚR

Cílem projektu bylo vytvořit backend pro webové aplikace umožňující pokročilou autentizaci a administraci.

Výsledkem snažení je funkční webová aplikace s možností přihlášení a registrace skrze aplikace třetích stran, úpravou profilu a administrací. Vše je ukázáno na prostém příkladu použití.

Jediným nespěným cílem a zároveň případnou úpravou do budoucna je sprovoznění aplikace online na webu. Aplikace je pro produkci však plně připravena.

Práce na tomto projektu mi umožnila využít mé znalosti z minulosti, zároveň jsem se toho však naučil mnoho nového a to se mi bude určitě hodit.

GitHub repozitář práce: <https://github.com/petr-micola/Zaverecna-prace>



## SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

- [1] *Co jsou to webové aplikace a dynamické webové stránky* [online]. [cit. 2024-01-13]. Dostupné z: <https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html>
- [2] *Úvod do Django frameworku a webových aplikací v Pythonu* [online]. [cit. 2024-01-13]. Dostupné z: <https://www.itnetwork.cz/python/django/uvod-do-django-frameworku-a-webovych-aplikaci-v-pythonu>
- [3] *Django Project Structure and File Structure* [online]. [cit. 2024-01-13]. Dostupné z: <https://techvidvan.com/tutorials/django-project-structure-layout/>
- [4] *Django MVT* [online]. [cit. 2024-01-13]. Dostupné z: <https://www.javatpoint.com/django-mvt>
- [5] *Django Create project and Install a modern design* [online]. [cit. 2024-01-13]. Dostupné z: [https://www.youtube.com/watch?v=9h9SC34tNNU&ab\\_channel=AppSeed](https://www.youtube.com/watch?v=9h9SC34tNNU&ab_channel=AppSeed)
- [6] *Python Django Social Authentication* [online]. [cit. 2024-01-13]. Dostupné z: [https://www.youtube.com/watch?v=RyB\\_wdEZh0w&ab\\_channel=CodeWithStein](https://www.youtube.com/watch?v=RyB_wdEZh0w&ab_channel=CodeWithStein)
- [7] *Dockerizing Django with Postgres, Gunicorn, and Nginx* [online]. [cit. 2024-01-13]. Dostupné z: <https://testdriven.io/blog/dockerizing-django-with-postgres-gunicorn-and-nginx/>
- [8] *Django* [online]. [cit. 2024-01-13]. Dostupné z: <https://www.djangoproject.com/>
- [9] *Django-allauth* [online]. [cit. 2024-01-13]. Dostupné z: <https://docs.allauth.org/en/latest/>
- [10] *PostgreSQL* [online]. [cit. 2024-01-13]. Dostupné z: <https://www.postgresql.org/docs/>
- [11] *Docker* [online]. [cit. 2024-01-13]. Dostupné z: <https://docs.docker.com/>
- [12] *Gunicorn* [online]. [cit. 2024-01-13]. Dostupné z: <https://docs.gunicorn.org/en/stable/>

[13] *Nginx* [online]. [cit. 2024-01-13]. Dostupné z: <https://docs.nginx.com/>

[14] *Soft UI Dashboard* [online]. [cit. 2024-01-13]. Dostupné z: <https://www.creative-tim.com/learning-lab/bootstrap/overview/soft-ui-dashboard>