# ChatBox - Backend

**Șandru Alexandru**
**SIC - 3711A**

# Cuprins

# 1 Mediu de dezboltare

## 1.1 Cerințe de sitem

Sistem de operare: *Arch Linux*
Kernel: *6.17.2-arch1-1*
Manager de pachete: *pacman*

# 2 Configurarea mediului backend

## 2.1 Actualizarea Sistemului

Actualizați sistemul folosind comanda:
*sudo pacman -Syu*

## 2.2 Instalarea Python

Instalați Python folosind comanda:
*sudo pacman -S python*

## 2.3 Crearea și activarea unui mediu virtual

Creați mediul virtual în directorul proiectului folosind comanda:
*python -m venv .venv*

Activați mediul virtual folosind comanda:
*source venv/bin/activate*

**Extra:**
Pentru a dezactiva mediul virtual folosiți comanda: *deactivate*

## 2.4 Instalarea framework-ului Django

Instalați Django folosind comanda:
*pip install django*

## 2.5 Crearea proiectului Django

Creați un proiect nou Django numit backend folosind comanda:
*django-admin startproject backend*

Intrați în directorul proiectului folosind comanda:
*cd backend*

Rulați serverul pentru a verifica dacă funcționează folosind comanda:
python manage.py runserver

## 2.6 Configurarea conexiunii la baza de date

Instalați driverul MySQL pentru Python folosind comanda:
pip install mysqlclient

Instalați pachetul python-dotenv pentru a folosi fisiere env folosind comanda:
pip install python-dotenv

Creați fișierul .env în directorul rădăcină al proiectului care sa contina urmatoarele:
DB_NAME=chatbox_db
DB_USER=chatbox_db
DB_PASSWORD=chatbox_db
DB_HOST=127.0.0.1
DB_PORT=3306

Editați fișierul backend/settings.py și modificați secțiunea DATABASES astfel:

```
import os
from dotenv import load_dotenv
load_dotenv()
BASE_DIR = Path(__file__).resolve().parent.parent
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": os.getenv("DB_NAME"),
        "USER": os.getenv("DB_USER"),
        "PASSWORD": os.getenv("DB_PASSWORD"),
        "HOST": os.getenv("DB_HOST"),
        "PORT": os.getenv("DB_PORT"),
    }
}
```

## 2.7 Crearea aplicației principale

Creați aplicația principală numită api folosind comanda:
*python manage.py startapp api*

Adăugați aplicația în INSTALLED_APPS din backend/settings.py astfel:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'api',
]
```

## 2.8 Crearea migrărilor

Creeaza migrațiile folosind comanda:
*python manage.py makemigrations*

Aplicarea migrațiilor în baza de date folosind comanda:
*python manage.py migrate*

## 2.9 Generarea modelelor din baza de date existentă

Generați automat modelele din baza de date conectată folosind comanda:
*python manage.py inspectdb > models.py*

**Explicație comandă:**

- *python manage.py* > fișierul principal de administrare al proiectului Django (permite rularea comenzilor Django)
- *inspectdb* > citește structura bazei de date și generează modelele corespunzătoare tabelelor existente
- > > redirecționează rezultatul comenzii într-un fișier
- *models.py* > fișierul unde se salvează modelele generate

## 2.10 Eager Loading și Lazy Loading

Explicație teoretică:

- Lazy Loading  > relațiile dintre modele (`ForeignKey`, `ManyToMany`) nu sunt încărcate până când sunt accesate. Este comportamentul implicit al Django ORM.
- Eager Loading > relațiile sunt încărcate imediat împreună cu obiectul principal pentru a reduce numărul de interogări în baza de date.

Exemple de cod Django:

```
# Lazy Loading (implicit)
files = Files.objects.all()
for f in files:
    print(f.user.username)  # aici se face query separat pentru fiecare user

# Eager Loading
files = Files.objects.select_related('user').all()
for f in files:
    print(f.user.username)  # query-ul pentru user este deja făcut
```