



Universitatea Ștefan cel Mare din Suceava
Facultatea de Inginerie Electrică și Știința Calculatoarelor
Domeniul: Calculatoare și Tehnologia Informației
Program de studii: Știința și Ingineria Calculatoarelor

ChatBox – Aplicație web cu conversație LLM limitată și opțiuni de scalabilitate.

Masterand
Alexandru Șandru

Cuprins

1 Proiectare.....	3
1.1 Paradigme utilizate.....	3
1.2 De ce au fost alese aceste paradigme.....	3
1.3 Arhitectura aplicației.....	3
2 Implementare.....	5
2.1 Business layer.....	5
2.2 Librării suplimentare utilizate.....	5
2.3 Secțiuni de cod sau abordări deosebite.....	6
3 Utilizare.....	7
3.1 Pașii de instalare.....	7
3.2 Mod de utilizare.....	8

1 Proiectare

1.1 Paradigme utilizate

Proiectul este structurat în patru aplicații Django, fiecare utilizând o paradigmă diferită pentru a experimenta cu conceptele învățate la laborator:

- **api-orm-sf** – ORM Schema First: aplicație existentă și funcțională, neutilizată de aplicația principală. Tabelele sale sunt separate și notate cu „_old”, iar doar operațiunea GET este implementată. Are un meniu specific care redirecționează către endpointurile sale pentru a facilita testarea.
- **api-orm-cf** – ORM Code First: aplicație existentă și funcțională, care conține tabelele utilizate de aplicația principală. Operațiunea GET este implementată. Dispune de un meniu dedicat pentru fiecare endpoint, pentru o navigare ușoară.
- **api-mvc** – MVC: folosită pentru partea de administrare a aplicației, cu CRUD implementat pentru toate tabelele (excepând unul). MVC-ul este realizat custom, iar FK-urile permit selectarea datelor din alte tabele. Pentru a accesa aplicația este necesară autentificarea cu un cont de administrator; dacă utilizatorul nu este autentificat, meniurile și endpointurile nu pot fi accesate.
- **api** – API: backend-ul principal al aplicației care este realizat folosind Django REST Framework, iar frontend-ul (React + Vite) interacționează cu această aplicație pentru toate funcționalitățile utilizatorului. Fiecare endpoint este accesibil printr-un meniu dedicat, pentru navigare rapidă și clară.

1.2 De ce au fost alese aceste paradigme

Structura proiectului combină mai multe paradigme pentru a întări cunoștințele de programare și a înțelege avantajele și limitările fiecărei abordări. Alegerea lor a urmărit dezvoltarea unei baze solide într-un limbaj de programare (Python) și familiarizarea cu diferite modele utilizate în dezvoltarea aplicațiilor web.

1.3 Arhitectura aplicației

Aplicația are o arhitectură modulară, în care componentele interacționează pentru a asigura funcționalitatea completă:

- **Frontend (React + Vite):** gestionează interfața utilizatorului. Prin frontend, utilizatorul poate interacționa cu aplicația ChatBox pentru: crearea contului, autentificare, vizualizarea istoricului

conversațiilor, gestionarea documentelor încărcate pentru LLM și schimbarea planului/abonamentului.

- **Plan gratuit:** acces la LLM GPT-OSS cu un număr limitat de interacțiuni și fără posibilitatea de a încărca documente.
- **Plan premium:** acces nelimitat la LLM LLaMA 3.2 și la GPT-OSS conform limitelor abonamentului, cu posibilitatea de a încărca documente pentru conversații.
- **Backend (Django + Django REST Framework):** furnizează toate endpointurile necesare pentru funcționalitățile aplicației, gestionează logica de business, autentificarea și autorizarea utilizatorilor, precum și administrarea datelor. Backend-ul servește ca suport pentru frontend și pentru aplicațiile secundare (api-orm-sf, api-orm-cf, api-mvc), dar funcționalitatea principală este realizată prin API-ul aplicației api.
- **Baza de date (MySQL / MariaDB):** stochează datele aplicației, inclusiv tabelele pentru utilizatori, istoricul conversațiilor, documentele încărcate, abonamentele și alte informații legate de funcționarea ChatBox.
- **Modulele api-mvc, api-orm-cf, și api-orm-sf:** există pentru laboratoare și testare; frontend-ul accesează doar aplicația principală api. Aplicația api-mvc necesită autentificare cu cont de administrator pentru a accesa funcționalitățile sale.

2 Implementare

2.1 Business layer

Business layer reprezintă logica principală a aplicației, adică modul în care datele și funcționalitățile sunt procesate înainte de a fi trimise către frontend. În cazul aplicației ChatBox:

- Gestionarea autentificării utilizatorilor și a sesiunilor.
- Gestionarea abonamentelor și planurilor.
- Gestionarea conversațiilor cu LLM.
- Validarea și procesarea documentelor încărcate pentru a fi folosite de LLM.
- CRUD pentru datele aplicației (tabelele principale ale aplicației api și, parțial, api-mvc).

2.2 Librării suplimentare utilizate

În proiect au fost folosite următoarele librării principale pentru a facilita dezvoltarea aplicației:

- **Backend (Python/Django):**
 - **Django și Django REST Framework** – pentru crearea backend-ului și a API-urilor.
 - **mysqlclient** – pentru conectarea la baza de date MySQL/MariaDB.
 - **django-cors-headers** – pentru gestionarea cererilor cross-origin din frontend.
 - **pdfminer.six, pdfplumber, pypdfium2, Pillow** – pentru procesarea și manipularea fișierelor PDF și imagini încărcate de utilizatori.
 - **python-dotenv** – pentru gestionarea fișierelor sensibile/importante.
 - **ollama** – pentru integrarea LLM-ului.
- **Frontend (React + Vite):**
 - **React și React DOM** – pentru construirea interfeței utilizatorului.
 - **React Router DOM** – pentru navigarea între secțiuni.
 - **Recharts** – pentru vizualizarea datelor și grafice.
 - **Vite** – pentru un mediu de dezvoltare rapid și build modern.

2.3 Secțiuni de cod sau abordări deosebite

Unul dintre elementele centrale ale aplicației este endpointul de chat implementat în **ChatView**. Acesta gestionează interacțiunea utilizatorului cu modelele LLM (GPT-OSS 20B și LLaMA 3.2) și integrează funcționalitățile de business layer, planuri și context suplimentar din fișierele trimise de utilizator.

Abordări cheie:

1. Gestionarea conversațiilor și planurilor:

- Endpointul verifică utilizatorul și planul activ pentru a decide ce model LLM poate fi folosit.
- LLM-ul GPT-OSS are limită de utilizare în funcție de abonament, iar LLaMA 3.2 poate fi folosit nelimitat în planul premium.
- Istoricul conversațiilor este păstrat și inclus ca context în promptul trimis către LLM.

2. Integrarea documentelor utilizatorului:

- Funcția `get_user_files_content` permite includerea conținutului fișierelor .txt și .pdf încărcate de utilizator în contextul conversației LLM.
- Funcționalitatea de încărcare fișiere este disponibilă doar pentru utilizatorii premium, oferind context suplimentar pentru modelul AI.

3. Rulare model LLM:

- Modelul este selectat din **MODEL_MAPPING** în funcție de plan și limitările abonamentului.
- Se generează răspunsul LLM folosind biblioteca ollama, iar rezultatul este salvat în baza de date pentru următoarele conversații.

4. Structurarea răspunsului:

- Răspunsul include textul generat de LLM, informații despre mesajul salvat, modelul utilizat și numărul de interacțiuni rămase pentru GPT-OSS.

3 Utilizare

3.1 Pașii de instalare

1. Instalare și configurare pentru programator

Clonează repository-ul proiectului: `git clone https://github.com/Xhadrines/ChatBox.git`

a. Backend (Django)

- Navighează în directorul *backend*.
- Creează fișierul *.env* cu datele de conectare la baza de date:
 - `DB_NAME=[DB_NAME]`
 - `DB_USER=[DB_USER]`
 - `DB_PASSWORD=[DB_PASSWORD]`
 - `DB_HOST=[DB_HOST]`
 - `DB_PORT=[DB_PORT]`

NOTE: Înlocuiește valorile din paranteze cu cele corespunzătoare.

- Creează un mediu virtual: `python -m venv .venv`
- Activează mediul virtual: `source ./venv/bin/activate`
- Instalează dependențele: `pip install -r requirements.txt`
- Creează migrațiile: `python manage.py makemigrations`
- Aplică migrațiile: `python manage.py migrate`
- Instalează Ollama: <https://ollama.com/>
- Instalează modelul LLM gpt-oss:20b: `ollama pull gpt-oss:20b`
- Instalează modelul LLM llama3.2: `ollama pull llama3.2`
- Pornește backend-ul: `python manage.py runserver`
- Backend-ul rulează implicit pe <http://127.0.0.1:2307/>.

b. Frontend (React + Vite)

- Navighează în directorul *frontend*.
- Creează fișierul *.env* cu link-ul către backend:

VITE_CHAT_API=http://<IP_BACKEND>:<PORT_BACKEND>

NOTE: Înlocuiește <IP_BACKEND> și <PORT_BACKEND> cu valorile corespunzătoare pe care le obții de la backend.

- Instalează dependențele: *npm install*
- Pornește frontend-ul: *npm run dev*
- Aplicația rulează implicit pe <http://127.0.0.1:2002/>.

2. Instalare și configurare pentru beneficiar

Beneficiarul trebuie să urmeze pașii menționați anterior, dar poate folosi fișierele *.env* deja configurate și să ruleze backend-ul și frontend-ul local sau pe un server.

Este necesară instalarea Ollama și a modelelor LLM, deoarece aplicația ChatBox le folosește pentru generarea răspunsurilor.

3.2 Mod de utilizare

1. Accesează aplicația în browser la adresa frontend-ului (<http://127.0.0.1:2002/>).
2. Creează un cont sau autentifică-te dacă există deja unul.
3. Funcționalități principale:
 - Trimiterea mesajelor către LLM și primirea răspunsurilor.
 - Vizualizarea istoricului conversațiilor.
 - Gestionarea datelor personale și a planului/abonamentului.
 - Încărcarea documentelor pentru a fi utilizate în conversațiile cu LLM (disponibil doar în planul premium).
4. Administratorii pot folosi panoul de administrare disponibil în backend (api-mvc) sau în frontend pentru gestionarea aplicației și a utilizatorilor.