<Chatbot AI pentru un magazin online de adidasi >

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Sistemul propus este un **chatbot inteligent** pentru un **magazin online de sneakeri**, care oferă asistență automată utilizatorilor interesați de produse. Acesta are capacitatea de a înțelege întrebări formulate în limbaj natural și de a răspunde cu informații relevante despre produse: brand, model, culoare, mărimi disponibile, preț și stoc.

Chatbotul este construit pe baza unui backend dezvoltat în Python folosind **FastAPI**, iar pentru procesarea întrebărilor utilizatorilor utilizează biblioteca **spaCy** pentru **NLP (Natural Language Processing)**. Datele despre produse sunt stocate într-un fișier CSV care servește drept sursă principală de informații.

Interacțiunea cu utilizatorii se face prin intermediul unei interfețe simple (browser sau terminal), unde aceștia pot introduce întrebări precum „Mai aveți Nike Air Max 270 negru mărimea 42?”. Chatbotul va analiza întrebarea și va căuta în baza de date produsul corespunzător, oferind un răspuns rapid și structurat.

Scopul principal al aplicației este de a automatiza procesul de informare al clientului, reducând volumul de muncă al personalului uman și îmbunătățind experiența de navigare într-un magazin online.

### Scopul documentului

Scopul acestui document este de a descrie modul în care chatbotul inteligent pentru magazinul online de sneakeri va fi proiectat și implementat. Sunt detaliate componentele software utilizate, structura aplicației, modul de procesare a întrebărilor utilizatorilor și interacțiunea cu baza de date locală.

Documentul urmărește să prezinte toate deciziile de proiectare necesare pentru dezvoltarea chatbotului, de la alegerea tehnologiilor și până la structura codului sursă. Acesta servește drept ghid pentru implementare, testare și validarea funcționalităților sistemului.

## Prezentare generală și abordări de proiectare

### Prezentare generală

Proiectul constă în dezvoltarea unui **chatbot inteligent pentru un magazin online de sneakeri**, capabil să răspundă automat la întrebările clienților privind produsele disponibile. Acest chatbot folosește **procesarea limbajului natural (NLP)** pentru a înțelege cererile formulate de utilizatori în limbaj natural și pentru a extrage informațiile relevante (brand, model, mărime, culoare etc.).

Aplicația este construită folosind limbajul de programare **Python**, având ca tehnologii principale **FastAPI** pentru partea de backend și **spaCy** pentru procesarea NLP. Informațiile despre produse sunt stocate într-un fișier **CSV local**, iar căutările se realizează în timp real prin interogarea acestui fișier cu ajutorul bibliotecii **pandas**.

Chatbotul funcționează local, fără a depinde de internet sau de servicii externe. Utilizatorii pot accesa sistemul printr-un browser sau din consolă, trimițând întrebări către endpoint-ul API-ului, urmând ca sistemul să răspundă într-un format clar și structurat. Proiectul este destinat să automatizeze procesul de suport și să îmbunătățească experiența clienților în magazinul online.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

 Se presupune că utilizatorii vor introduce întrebări formulate corect gramatical, într-un limbaj natural simplu.

 Se presupune că fișierul sneakeri.csv este disponibil, completat corect și se află în același director cu aplicația.

 Se presupune că toate bibliotecile necesare (FastAPI, spaCy, pandas, uvicorn etc.) sunt instalate în mediul local.

 Se presupune că utilizatorii nu vor adresa întrebări în alte limbi sau folosind abrevieri informale.

.

#### Constrângeri

* Aplicația funcționează doar local, fără acces la internet sau API-uri externe.
* Baza de date este limitată la un fișier CSV; nu se folosesc baze de date relaționale sau sisteme de gestiune complexe.
* NLP este implementat exclusiv cu spaCy și modelul en\_core\_web\_sm, ceea ce poate limita acuratețea în cazul unor întrebări complexe.
* Interfața utilizator este simplă (consolă sau browser), fără integrare grafică avansată.
* Sistemul nu gestionează conturi de utilizator, autentificare sau sesiuni persistente.

#### Riscuri

 Întrebările formulate ambiguu sau în mod neclar pot duce la **răspunsuri eronate** sau incomplete.

 **Fișierul CSV** poate deveni inutilizabil dacă este modificat greșit sau corupt manual.

 Modelul spaCy utilizat nu este antrenat special pentru limba română, ceea ce poate duce la **scăderea performanței NLP**.

 Lipsa unui sistem de jurnalizare/logging face dificilă **urmărirea erorilor** în cazul unor cereri neprocesate corect.

 Dacă aplicația nu este pornită corect (ex. lipsesc pachete sau fișiere), sistemul va eșua la rulare fără un mecanism de recuperare.

## Considerații de proiectare

### Obiective și linii directoare (ghiduri)

Obiectivul principal al sistemului este de a crea un **chatbot inteligent**, capabil să ofere răspunsuri automate și corecte clienților unui magazin online de sneakeri, folosind procesarea limbajului natural (NLP). Accentul în proiectare este pus pe:

* **Accesibilitate**: sistemul trebuie să fie ușor de folosit, chiar și de către utilizatori fără cunoștințe tehnice;
* **Performanță**: timpul de răspuns la interogări trebuie să fie sub 2 secunde;
* **Simplitate și modularitate**: aplicația este împărțită clar în module: analiză NLP, interfață API, căutare în fișierul CSV.

Designul este condus de principiul „keep it simple” – se evită soluțiile complexe și se folosesc tehnologii ușor de întreținut. Codul este organizat în funcții clare, comentat și respectă convențiile Python. De asemenea, chatbotul este proiectat să funcționeze **100% offline**, fără dependențe de rețea sau baze de date externe.

### Metode de dezvoltare

Pentru dezvoltarea sistemului chatbot s-a adoptat o metodologie **iterativă și incrementală**, cu accent pe testarea continuă și livrarea funcționalităților în pași clari. Procesul de dezvoltare a fost împărțit în etape distincte, după cum urmează:

1. **Analiza cerințelor** – definirea funcționalităților esențiale: răspuns la întrebări despre produse, verificarea stocului, afișarea detaliilor.
2. **Proiectarea arhitecturii** – structurarea aplicației în componente independente: backend FastAPI, procesare NLP cu spaCy, citire date cu pandas.
3. **Implementare modulară** – dezvoltarea și testarea separată a fiecărui modul, pentru a permite depanarea rapidă și integrarea ușoară.
4. **Testare locală** – testarea chatbotului cu diverse întrebări formulate natural, pentru a verifica robustetea interpretării și generarea de răspunsuri.
5. **Îmbunătățiri și optimizări** – extinderea listei de expresii recunoscute, rafinarea scorurilor de potrivire, gestionarea excepțiilor și formularea de răspunsuri prietenoase.

Proiectul a fost dezvoltat și testat într-un mediu local, folosind Python 3.10 și editorul de cod Visual Studio Code. Toate componentele au fost gândite astfel încât să poată fi reutilizate și extinse ușor într-o aplicație reală de e-commerce.

### Strategii de arhitectură

* Arhitectura aplicației se bazează pe un model **client-server** simplu, cu un **backend local** care răspunde la întrebările utilizatorilor printr-un API REST, utilizând **FastAPI**. Aplicația nu depinde de baze de date externe, iar datele despre produse sunt stocate într-un fișier CSV local, ceea ce face aplicația mai ușor de implementat și de întreținut.
* **Principalele strategii adoptate** includ:

1. **Modularitatea**: Aplicația este împărțită în module separate, fiecare având o funcționalitate clară (de exemplu, procesarea NLP cu spaCy, gestionarea fișierului CSV cu pandas, API-ul FastAPI pentru interacțiunea cu utilizatorul). Această structură modulară permite actualizări rapide și modificări în funcție de cerințele viitoare.
2. **Scalabilitate**: Deși aplicația este inițial implementată pentru a rula local, arhitectura permite integrarea ușoară cu baze de date externe sau API-uri externe, în cazul în care sistemul crește și necesită mai multe surse de date sau funcționalități avansate.
3. **Descentralizarea**: Procesarea întrebărilor și generarea răspunsurilor se face local pe serverul FastAPI, fără dependențe externe. Acest lucru oferă un control total asupra datelor și reduce riscurile de performanță legate de accesul la rețea sau servere externe.
4. **Performanță**: Pentru a asigura răspunsuri rapide la întrebările utilizatorilor, aplicația utilizează **indexare optimizată** a fișierului CSV cu ajutorul **pandas** și **procesare NLP rapidă** cu spaCy. Timpul de răspuns al sistemului este menținut sub 2 secunde pentru majoritatea întrebărilor.
5. **Securitate**: Comunicarea între client și server este criptată folosind **HTTPS** sau **TLS** pentru a asigura confidențialitatea și integritatea datelor. De asemenea, se implementează măsuri de securitate pentru a proteja datele sensibile ale utilizatorilor și pentru a preveni atacurile cibernetice, cum ar fi **SQL injection** sau **XSS**.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Aplicația este construită pe un model **client-server** simplu. Componentele principale sunt:

1. **Frontend**: Permite utilizatorului să adreseze întrebări prin browser sau terminal.
2. **Backend (FastAPI)**: Procesează cererile utilizatorilor și returnează răspunsurile.
3. **NLP (spaCy)**: Analizează întrebările și extrage entitățile relevante.
4. **Fișier CSV**: Conține datele despre produse, accesate prin **pandas**.

Componentele colaborează pentru a permite răspunsuri rapide la întrebările utilizatorilor. Alegerea unui fișier CSV local pentru stocarea datelor permite o soluție simplă și rapidă, fără dependență de baze de date externe.

Arhitectura a fost aleasă pentru modularitate, performanță și ușurință în întreținere. **FastAPI** asigură o performanță rapidă, iar **spaCy** este utilizat pentru procesarea limbajului natural. Această decompunere permite extinderea ușoară a aplicației pe măsură ce cerințele evoluează.

### 4.1 Vedere logică

Aplicația este împărțită în trei componente principale:

1. **Frontend**: Utilizatorii trimit întrebări printr-o interfață simplă (browser sau terminal).
2. **Backend (FastAPI)**: Procesează cererile, analizează întrebările cu **spaCy** și interoghează fișierul CSV.
3. **Fișier CSV**: Conține datele despre produse, gestionate cu **pandas**.

Răspunsurile sunt trimise utilizatorilor în format text sau JSON.

### 4.2 Arhitectură hardware

Sistemul este **centralizat**, rulând pe un singur server local, fără distribuirea procesării între mai multe instanțe. Hardware-ul utilizat include următoarele componente:

1. **Serverul principal (Aplicație și Date)**:
   * **Tip**: Server desktop sau server dedicat
   * **Procesor**: Intel Core i5 sau echivalent
   * **Memorie RAM**: 8 GB
   * **Stocare**: 100 GB SSD (pentru a gestiona fișierele CSV și datele operaționale)
   * **Locație**: Locat într-un mediu local (pe mașina de dezvoltare sau pe un server dedicat)
   * **Rol**: Aici rulează aplicația FastAPI, motorul NLP (spaCy), precum și fișierul CSV cu datele produselor.
2. **Firewall**:
   * **Tip**: Configurat pentru a proteja aplicația de accesul neautorizat.
   * **Locație**: Instalarea pe serverul de aplicație, la nivel de rețea.
   * **Rol**: Blochează traficul neautorizat și protejează porturile aplicației.
3. **Componente periferice**:
   * **Routere**: Pentru gestionarea traficului de rețea intern.
   * **Switchuri**: Pentru conectivitate internă între servere și dispozitive.
   * **Load Balancer** (opțional, dacă se adaugă un sistem distribuit pe viitor): Pentru distribuirea traficului între mai multe servere.

### 4.3 Arhitectură software

Arhitectura software a aplicației este bazată pe un model **client-server**, în care logica aplicației este distribuită între **frontend**, **backend** și **baza de date**. Iată componentele principale:

1. **Frontend (Interfața utilizatorului)**:
   * **Tehnologie**: Interfața este simplă și poate fi implementată printr-un browser web sau un terminal.
   * **Limbaj de programare**: HTML, CSS, JavaScript (pentru interfața web, dacă este utilizată).
   * **Rol**: Permite utilizatorilor să adreseze întrebări despre produsele din magazin și să vizualizeze răspunsurile generate de sistem.
2. **Backend (FastAPI)**:
   * **Tehnologie**: FastAPI este folosit pentru a construi API-ul backend al aplicației.
   * **Limbaj de programare**: Python 3.10+
   * **Funcție**: Procesează cererile utilizatorilor, interacționează cu motorul NLP și cu baza de date (fișierul CSV), iar la final trimite răspunsul utilizatorului.
   * **Interfețe API**: Comunicarea între frontend și backend se face prin intermediul cererilor HTTP (REST API).
3. **Motor NLP (spaCy)**:
   * **Tehnologie**: **spaCy** este utilizat pentru procesarea limbajului natural, cu modelul en\_core\_web\_sm pentru a analiza întrebările utilizatorilor.
   * **Funcție**: Extrage entitățile (brand, model, mărime etc.) din întrebările formulate în limbaj natural, pentru a putea interoga baza de date corect.
4. **Baza de date (Fișier CSV)**:
   * **Tehnologie**: Datele despre produse sunt stocate într-un fișier CSV, gestionat de biblioteca **pandas**.
   * **Funcție**: Conține informații despre produse, precum brand, model, mărime, preț, descriere și stoc. Aceste date sunt accesate și filtrate în funcție de cerințele utilizatorilor.
5. **Interacțiuni între componente:**

* **Frontend** trimite întrebările utilizatorilor către **Backend (FastAPI)**.
* **Backend** folosește **spaCy** pentru analiza întrebării și extragerea entităților.
* **Backend** interoghează fișierul **CSV** folosind **pandas** pentru a căuta produsul solicitat.
* **Răspunsul** este trimis înapoi utilizatorului prin **Frontend**.

### 4.4 Arhitectura informațiilor

Arhitectura informațiilor din cadrul aplicației chatbot pentru magazinul de sneakeri este construită pentru a asigura gestionarea eficientă a datelor și procesarea acestora în mod rapid și simplu. În această arhitectură, **fișierul CSV** servește ca sursă principală de date, iar informațiile despre produse sunt accesibile și procesate local.

Aplicația utilizează un **flux clar de date**:

* Utilizatorul trimite întrebarea prin intermediul frontend-ului (browser sau terminal).
* Backend-ul, implementat cu **FastAPI**, primește întrebarea și folosește **spaCy** pentru a analiza întrebarea utilizatorului și a extrage entitățile relevante, precum brandul, modelul, culoarea și mărimea.
* După extragerea entităților, backend-ul interoghează **fișierul CSV** folosind **pandas** pentru a căuta produsele relevante.
* Răspunsul, care include informațiile corespunzătoare despre produs (de exemplu, preț, stoc, descriere), este trimis utilizatorului prin frontend.

### 4.5 Arhitectura de comunicații interne

Arhitectura de comunicații interne a aplicației este bazată pe un model client-server, în care diferitele componente software comunică între ele prin intermediul protocoalelor HTTP/REST. Comunicarea începe atunci când utilizatorul trimite o întrebare prin frontend (browser sau terminal) către backend-ul FastAPI. Backend-ul preia întrebarea și o trimite către motorul de procesare NLP (spaCy), care analizează întrebarea și extrage entitățile relevante, precum brandul, modelul sau mărimea.

După ce motorul NLP a identificat entitățile, backend-ul interoghează fișierul CSV, care conține informațiile despre produsele disponibile, folosind biblioteca pandas. Backend-ul filtrează aceste date și pregătește răspunsul pentru utilizator, care este apoi trimis înapoi prin frontend. Răspunsul poate fi furnizat sub formă de text simplu sau în format JSON, în funcție de configurația aplicației.

### 4.6 Diagrama de arhitectură a sistemului

+--------------------+

| Utilizator |

| (Browser/Terminal) |

+---------+----------+

|

| Întrebare (HTTP)

v

+--------------------------+

| Backend |

| (FastAPI - API) |

+-----------+--------------|

| Analiză NLP

v

+--------------------------+

| Procesare NLP (spaCy) |

+-----------+--------------+

| Entități extrase

v

+--------------------------+

| Fișier CSV |

| (Date despre produse) |

+-----------+--------------+

| Date filtrate

v

+--------------------------+

| Răspuns generat (JSON) |

+-----------+--------------

| Răspuns HTTP

v

+--------------------+

| Utilizator |

+--------------------+

## 5. Proiectarea sistemului

### Proiectarea bazei de date

Pentru acest proiect, baza de date este implementată sub forma unui fișier CSV care acționează ca o bază de date statică, locală. Alegerea unui fișier CSV a fost motivată de simplitatea structurii aplicației, rularea locală și lipsa unei nevoi reale pentru un sistem de gestiune a bazelor de date relaționale (SGBD).

Fișierul CSV conține datele necesare chatbotului pentru a oferi răspunsuri utilizatorilor. Structura acestuia este tabelară, fiecare linie reprezentând un produs disponibil în magazinul online. Coloanele sunt echivalente cu atributele produselor și pot fi considerate drept câmpuri ale unei baze de date relaționale simple.

Fișierul este procesat cu biblioteca pandas, care permite citirea rapidă a datelor, filtrarea și căutarea produselor în funcție de entitățile extrase din întrebările utilizatorilor.

Această abordare oferă flexibilitate pentru proiecte de dimensiuni reduse și permite actualizarea ușoară a datelor prin editarea manuală a fișierului.

**Obiecte de date și structuri de date rezultante**

În cadrul aplicației, datele despre produse sunt organizate sub formă de obiecte și structuri de date procesate intern de către sistem, pornind de la fișierul CSV.

**Obiectele de date** principale sunt reprezentate de fiecare produs din magazin. Fiecare rând din fișierul sneakeri.csv este interpretat ca un obiect de tip „produs”, cu următoarele atribute:

* id: identificator unic
* brand: marca produsului (ex. Nike)
* model: denumirea modelului (ex. Air Force 1)
* culoare: culoarea principală a produsului
* marime: dimensiunea pantofului sport (ex. 42)
* pret: prețul în lei
* descriere: informații suplimentare
* stoc: cantitatea disponibilă în stoc

La nivel de cod, fișierul CSV este citit și transformat într-o **structură de tip DataFrame** oferită de biblioteca pandas. Acest DataFrame este structura de bază pe care aplicația o utilizează pentru filtrarea, sortarea și căutarea produselor pe baza entităților extrase cu ajutorul procesării NLP.

După ce utilizatorul adresează o întrebare, rezultatele obținute în urma procesării sunt convertite în obiecte Python de tip dict, care sunt apoi serializate (de exemplu, în format JSON) și trimise ca răspuns utilizatorului.

Această organizare permite o manipulare eficientă a datelor și oferă flexibilitate în extragerea, afișarea și modificarea conținutului bazei de date statice.

**Fișiere și baze de date**

Sistemul nu utilizează un sistem de baze de date relațional (SGBD) clasic, ci operează pe baza unui **fișier de tip CSV**, care acționează ca o bază de date locală statică. Acest fișier se numește sneakeri.csv și este plasat în același director cu aplicația. El conține toate informațiile necesare despre produsele disponibile în magazinul online.

Fișierul sneakeri.csv este structurat în mod tabelar, având o linie de antet care definește câmpurile (coloanele), urmată de rânduri care conțin datele fiecărui produs. Fișierul este gestionat prin intermediul bibliotecii **pandas** din Python, care oferă funcționalități avansate pentru citire, filtrare, sortare și extragere de informații.

Accesul la date se face **doar în citire**, aplicația nepermițând modificarea fișierului CSV prin intermediul interfeței utilizatorului. Orice modificare se face manual, în afara aplicației.

Acest model de stocare este potrivit pentru aplicații mici sau prototipuri, precum acest proiect de licență, și oferă simplitate, flexibilitate și un timp de acces foarte scurt asupra datelor, fără a introduce complexitatea suplimentară asociată bazelor de date relaționale sau cloud.

**Baze de date**

Aplicația nu utilizează un sistem clasic de baze de date relaționale (precum MySQL, PostgreSQL sau SQLite), ci se bazează pe un **fișier de tip CSV** ca sursă principală de date. Acest fișier, denumit sneakeri.csv, joacă rolul unei baze de date statice, fiind ușor de gestionat și potrivit pentru aplicații de dimensiuni mici și rulare locală.

Structura acestui fișier corespunde unei tabele, în care fiecare rând reprezintă un produs, iar fiecare coloană corespunde unui atribut (brand, model, culoare, marime, preț, stoc etc.). Aplicația încarcă acest fișier în memorie folosind biblioteca **pandas**, transformându-l într-un **DataFrame** – o structură de date optimizată pentru lucrul cu tabele mari de date.

**Fișiere non-DBMS**

În cadrul aplicației chatbot pentru magazinul online de sneakeri, fișierul sneakeri.csv este utilizat ca sursă de date statică, **în afara unui sistem de gestiune a bazelor de date (DBMS)**. Acest fișier este considerat un **fișier non-DBMS**, deoarece nu implică un server de baze de date, un motor de interogare SQL sau funcționalități de tranzacționare ori gestionare a utilizatorilor.

Fișierul este de tip **CSV (Comma Separated Values)** și conține toate înregistrările necesare despre produsele disponibile în aplicație. Structura sa este simplă și constă dintr-un antet (care definește câmpurile) și rânduri de date (care descriu produsele).

### Conversii de date

Conversiile de date reprezintă un proces esențial în funcționarea aplicației, permițând transformarea datelor dintr-o formă brută, neprocesată, într-o formă inteligibilă și relevantă pentru utilizator. În cadrul acestui sistem, conversiile se realizează în principal în două direcții: de la textul introdus de utilizator către structuri de date interne și de la date brute către un răspuns structurat.

În prima etapă, întrebarea formulată de utilizator este preluată ca text liber și analizată cu ajutorul motorului **NLP (spaCy)**. Această analiză implică conversia întrebării într-un set de entități extrase: brand, model, culoare, mărime etc. Astfel, limbajul natural este transformat într-un format logic, procesabil de sistem.

După extragerea entităților, se realizează o altă conversie: identificarea și filtrarea datelor relevante din fișierul sneakeri.csv. Acest fișier este încărcat într-un obiect de tip **DataFrame** folosind biblioteca pandas, iar rezultatul interogării este un rând sau un set de rânduri corespunzătoare cererii utilizatorului.

În final, informațiile filtrate sunt convertite într-un **obiect Python de tip dicționar** și apoi într-un răspuns text sau JSON, lizibil pentru utilizator. Această conversie asigură o comunicare eficientă între utilizator și aplicație, indiferent de forma în care sunt stocate datele intern.

.

### Interfețe utilizator

Interfața utilizatorului este concepută pentru a fi cât mai simplă și intuitivă, având rolul de a permite clienților să interacționeze rapid și ușor cu chatbotul pentru a obține informații despre produsele disponibile în magazinul online de sneakeri.

În forma sa de bază, interfața poate fi implementată fie sub formă de **linie de comandă (CLI)**, fie ca o **interfață web minimalistă**, accesibilă printr-un browser. În ambele cazuri, utilizatorul introduce întrebări în limbaj natural, cum ar fi „Mai aveți Nike Air Max 42 alb?”, iar sistemul răspunde cu un mesaj clar și structurat.

Interacțiunea este unidirecțională și intuitivă: utilizatorul introduce textul, serverul procesează cererea, extrage datele relevante și returnează un răspuns informativ. Răspunsurile pot include detalii despre produs (model, culoare, preț, mărime), precum și informații despre disponibilitatea în stoc.

Interfața este astfel proiectată încât să nu necesite autentificare sau navigare prin meniuri complexe. Totul se desfășoară într-un format de tip întrebare-răspuns, similar aplicațiilor de mesagerie, ceea ce face ca experiența utilizatorului să fie rapidă, ușor de înțeles și prietenoasă.

#### Intrări

#### Singura formă de intrare a utilizatorului în sistem este textul liber, introdus sub formă de întrebare. Utilizatorii scriu întrebări în limbaj natural, iar aplicația le interpretează automat prin motorul de procesare a limbajului natural (NLP). Exemple de intrări posibile includ:

#### „Mai aveți Nike Air Max 90 mărimea 42?”

#### „Ce adidași Adidas albi aveți pe stoc?”

#### „Cât costă Puma RS-X negru, 41?”

#### Aceste intrări sunt primite de către aplicație prin intermediul unei interfețe (web sau terminal) și sunt transmise către backend pentru procesare. Nu sunt necesare clicuri, formulare sau interacțiuni complexe – sistemul este complet controlat prin comenzi textuale, ceea ce îl face rapid și intuitiv.

#### Ieșiri

### Ieșirile sistemului constau în răspunsuri text generate automat de chatbot pe baza întrebărilor utilizatorului. Aceste răspunsuri sunt afișate direct în interfață (terminal sau browser) și conțin informații relevante extrase din baza de date locală (fișierul CSV).

### Răspunsurile pot include:

### Confirmarea existenței unui produs în stoc;

### Detalii despre produs: brand, model, culoare, mărime, preț;

### Notificarea lipsei unui produs în stoc;

### Recomandări în cazul în care produsul solicitat nu este găsit exact.

### Exemple de ieșiri:

### „Avem 2 bucăți disponibile din Nike Air Max 90 alb, mărimea 42, la prețul de 499 RON.”

### „Din păcate, produsul solicitat nu este în stoc.”

### „Adidas Superstar alb este disponibil în mărimile 40, 41 și 43.”

## 6.Scenarii de utilizare

Scenariile de utilizare descriu modul în care un utilizator interacționează cu chatbotul pentru a obține informații despre produsele din magazinul online de sneakeri. Sistemul este gândit pentru interacțiuni rapide, în limbaj natural, fără navigare sau autentificare.

**Scenariul 1 – Căutarea unui produs după brand, model și mărime**

**Utilizator**: Introduce întrebarea „Mai aveți Nike Air Max 270 mărimea 42?”  
**Sistem**: Analizează întrebarea, identifică brandul, modelul și mărimea, caută în fișierul CSV și răspunde:  
„Avem 3 bucăți disponibile din Nike Air Max 270, mărimea 42, culoare negru, preț 549 RON.”

**Scenariul 2 – Întrebare generală despre produse disponibile**

**Utilizator**: Scrie „Ce modele Adidas albe aveți pe stoc?”  
**Sistem**: Extrage brandul și culoarea, filtrează produsele corespunzătoare și returnează o listă cu articole disponibile: „Adidas Superstar alb – 499 RON, mărimile: 40, 41.  
Adidas ZX 700 alb – 469 RON, mărimea 42.”

**Scenariul 3 – Produsul nu este în stoc**

**Utilizator**: Scrie „Există Puma RS-X mărimea 43?”  
**Sistem**: Găsește produsul în baza de date, dar stocul este 0, și răspunde: „Din păcate, Puma RS-X mărimea 43 nu este în stoc.”

**Scenariul 4 – Informații despre preț**

**Utilizator**: Întreabă „Cât costă Nike Air Force 1 alb, mărimea 41?”  
**Sistem**: Răspunde:  
„Nike Air Force 1 alb, mărimea 41 – 529 RON. Disponibil: 2 bucăți.”

## 7.Proiectare de detaliu.

**Proiectarea de detaliu** se concentrează pe implementarea exactă a funcționalităților aplicației și interacțiunile dintre componentele acesteia. În cazul aplicației de chatbot pentru magazinul de sneakeri, designul include backend-ul, procesarea NLP, manipularea datelor și formatarea răspunsurilor.

### Proiectare hardware de detaliu

### Arhitectura hardware a aplicației este destul de simplă, deoarece sistemul rulează într-un mediu local, fără a depinde de infrastructuri complexe sau de resurse externe. Componentele hardware sunt împărțite în două mari categorii: serverul de aplicație și perifericele necesare pentru interacțiunea cu utilizatorul.

### Serverul de aplicație:

### Tip: Un server desktop sau un server dedicat pentru rularea aplicației.

### Specificații:

### Procesor: Intel Core i5 (sau echivalent) cu cel puțin 4 nuclee, pentru a asigura procesarea rapidă a cererilor utilizatorilor și a motorului NLP.

### Memorie RAM: 8 GB, suficient pentru a gestiona mai multe cereri simultane și pentru a permite încărcarea fișierului CSV în memorie.

### Stocare: SSD de 100 GB, suficient pentru fișierul CSV și alte date operaționale. Stocarea rapidă asigură accesul rapid la fișierele de date.

### Componente periferice:

### Routere și switch-uri: Pentru conectivitatea internă a rețelei, esențială pentru comunicarea între server și utilizator.

### Firewall: Este implementat la nivelul rețelei, pentru a asigura protecția împotriva accesului neautorizat la aplicație.

### Conectivitate și Securitate:

### Porturi: Aplicația utilizează un port local (de obicei portul 8000 pentru FastAPI) pentru a accepta cereri de la utilizatori.

### Securitate: Toate comunicațiile între utilizatori și server sunt protejate prin HTTPS, pentru a garanta confidențialitatea și integritatea datelor.

### Proiectare software de detaliu

Proiectarea software-ului de detaliu se referă la definirea și implementarea fiecărei componente software, interacțiunii dintre ele și modul în care acestea îndeplinesc funcționalitățile aplicației. În cadrul aplicației chatbot pentru magazinul de sneakeri, detaliile proiectării includ organizarea codului, modul în care sunt gestionate datele și implementarea procesării limbajului natural (NLP).

* Backend (FastAPI):
  + **Endpoint API**: Aplicația are un endpoint principal /chat/ care primește cereri de la utilizatori (intrebări în limbaj natural), le procesează și trimite răspunsuri. FastAPI este utilizat pentru a gestiona cererile HTTP și pentru a interacționa cu motorul NLP și fișierul CSV.
  + **Validare**: FastAPI include validare automată a datelor primite de la utilizatori, pentru a se asigura că întrebările sunt corect formate.
  + **Comunicare cu NLP**: Backend-ul trimite întrebările procesate către motorul NLP pentru a extrage entitățile relevante (brand, model, mărime), apoi interoghează fișierul CSV.
* Motorul NLP (spaCy):
  + **Analiza întrebărilor**: spaCy este folosit pentru procesarea limbajului natural, extrăgând entitățile din întrebările utilizatorilor. Aceste entități sunt apoi folosite pentru a căuta în fișierul CSV produsele corespunzătoare.
  + **Modelul**: Se folosește modelul en\_core\_web\_sm, care este suficient pentru a procesa întrebări simple legate de produsele din magazin.
* Fișierul CSV:
  + **Stocarea datelor**: Fișierul CSV este citit și procesat de backend cu ajutorul bibliotecii pandas. Acesta conține datele despre produsele disponibile, iar aplicația caută în mod eficient produsele care corespund criteriilor extrase din întrebările utilizatorilor.
  + **Interogarea datelor**: Pandas filtrează datele rapid, oferind o performanță bună la interogarea fișierelor mici sau medii.
* Răspunsuri și formatarea ieșirii:
  + **Generarea răspunsurilor**: După interogarea fișierului CSV și obținerea rezultatelor, backend-ul generează un răspuns care este trimis înapoi utilizatorului. Acesta poate include informații despre produs (de exemplu, preț, stoc, mărimi disponibile).
* Arhitectura modulară:
  + Codul este structurat în mod modular, fiecare componentă fiind responsabilă pentru o anumită funcționalitate. De exemplu, procesarea întrebărilor se face într-un modul dedicat NLP, iar manipularea datelor se face în altul.

### Proiectare detaliată de securitate

Instrucțiuni: Furnizați informații detaliate pentru fiecare dintre componentele individuale de securitate hardware. Specificați elementele de mai jos, după cum este necesar.

• Autentificare

• Autorizare

• Jurnalizare(arhivare) și auditare

• Criptare

• Utilizarea porturilor de rețea

• Detectare și prevenire a intruziunilor.

### 

### Proiectare de detaliu pentru performanța sistemului

Pentru asigurarea performanței ridicate a aplicației, s-au luat măsuri specifice în fiecare componentă a sistemului. **FastAPI** a fost ales pentru backend datorită vitezei mari de procesare a cererilor HTTP, asigurând un timp de răspuns rapid chiar și la un număr mare de cereri simultane. În plus, fișierul **CSV**, care servește drept bază de date locală, este citit o singură dată la pornirea aplicației și stocat în memorie sub forma unui **DataFrame pandas**. Această abordare permite interogarea rapidă a datelor, fără a solicita citirea fișierului la fiecare cerere, optimizând astfel timpii de acces.

Fișierul CSV este structurat astfel încât să poată fi accesat rapid, iar **pandas** furnizează operațiuni de filtrare și căutare eficiente. Acest lucru reduce timpul de acces la informațiile despre produse și ajută la menținerea performanței aplicației chiar și pe măsură ce volumul de date crește ușor.

Chiar dacă aplicația funcționează într-un mediu local, arhitectura este pregătită să susțină o scalabilitate viitoare, permițând integrarea unui sistem de baze de date relațional sau cloud, în cazul în care aplicația ar necesita procesarea unui număr mai mare de cereri sau date. În prezent, sistemul este capabil să răspundă rapid la întrebările utilizatorilor, fără întârzieri semnificative, iar **spaCy**, folosit pentru procesarea limbajului natural, contribuie la viteza ridicată de analiză a întrebărilor, fără a afecta performanța sistemului.

În ceea ce privește utilizarea resurselor, aplicația a fost proiectată pentru a consuma un număr minim de resurse. Fișierul CSV este încărcat o singură dată și procesat în mod eficient pentru fiecare interogare, astfel încât memoria utilizată rămâne constantă și nu crește exponențial pe măsură ce numărul cererilor crește.

### Proiectare detaliată a comunicațiilor interne (între componente)

1. Comunicarea între componentele aplicației este esențială pentru asigurarea unei interacțiuni fluide și rapide. În cazul aplicației de chatbot pentru magazinul de sneakeri, aceasta se face într-un mediu **client-server**, iar datele sunt transferate între **frontend**, **backend**, **motorul NLP (spaCy)** și **fișierul CSV**.
2. **Frontend-ul**, care poate fi o aplicație de terminal sau o interfață web, trimite întrebările utilizatorilor printr-un **request HTTP** către **backend-ul FastAPI**. Aceasta este o comunicare asincronă, asigurându-se că utilizatorul poate trimite cereri fără a bloca alte procese. Cererile HTTP sunt gestionate de backend, care, la rândul său, utilizează **spaCy** pentru procesarea limbajului natural al întrebării.
3. După ce motorul NLP extrage entitățile din întrebare (de exemplu, brand, model, mărime), backend-ul interoghează **fișierul CSV** pentru a găsi produsele care corespund criteriilor extrase. Acest proces de interogare a fișierului CSV se face cu ajutorul **pandas**, care permite manipularea rapidă a datelor și obținerea unui răspuns eficient.
4. După obținerea datelor din fișierul CSV, **backend-ul** formează un răspuns corespunzător, pe care îl trimite înapoi către **frontend** sub formă de **text sau JSON**, care este apoi afișat utilizatorului.
5. Comunicarea între backend și fișierul CSV se face local, fără a depinde de resurse externe, iar utilizarea protocoalelor **HTTP/HTTPS** garantează confidențialitatea și integritatea datelor transferate între componente.
6. Astfel, toate componentele interacționează eficient, fără a exista blocaje semnificative între procesarea cererilor și generarea răspunsurilor, iar aplicația este capabilă să răspundă rapid întrebărilor utilizatorilor.

## Controale pentru verificarea integrității sistemului

Pentru a asigura integritatea și corectitudinea aplicației, sistemul include mai multe controale și proceduri de validare a datelor și a proceselor interne. Aceste controale sunt esențiale pentru a preveni erorile și pentru a asigura că aplicația răspunde corect și eficient la cerințele utilizatorilor.

1. **Validarea întrebărilor utilizatorului**:
   * În momentul în care utilizatorul trimite o întrebare, backend-ul (FastAPI) validează inputul pentru a se asigura că textul introdus este conform cu formatul așteptat. De exemplu, backend-ul va verifica dacă întrebarea conține cuvinte cheie precum „brand”, „model” și „mărime”.
   * În cazul în care inputul nu este valid sau este ambigu, utilizatorului i se va trimite un mesaj de eroare clar, indicând ce informații lipsesc.
2. **Verificarea integrității fișierului CSV**:
   * Fișierul sneakeri.csv este încărcat la fiecare pornire a aplicației, iar integritatea acestuia este verificată. Sistemul asigură că toate coloanele sunt prezente și că datele nu sunt corupte.
   * În cazul în care fișierul CSV este modificat manual, aplicația include un control de integritate care semnalează utilizatorului orice eroare de format sau date lipsă.
3. **Monitorizarea performanței și a răspunsurilor**:
   * Sistemul monitorizează timpul de răspuns pentru fiecare cerere, iar în cazul în care timpul depășește un prag prestabilit, aplicația va semnala o eroare internă. Aceasta poate fi cauzată de încărcarea excesivă a resurselor sau de un blocaj în procesarea întrebării.
   * De asemenea, se utilizează loguri pentru a urmări fiecare cerere procesată și pentru a detecta eventualele erori în fluxul de date.
4. **Verificarea consistenței datelor**:
   * În fiecare moment, aplicația se asigură că datele extrase din fișierul CSV sunt consistente, iar valorile (precum prețul, mărimea și stocul) sunt corecte și nu sunt contradictorii. De exemplu, aplicația verifică dacă stocul unui produs este mai mare decât 0 atunci când acesta este disponibil pentru vânzare.
5. **Backup și recuperare în caz de eroare**:
   * Pentru a asigura continuitatea aplicației, un sistem de backup automat al fișierului CSV este implementat. În cazul în care fișierul este corupt sau există probleme în timpul procesării datelor, aplicația poate restaura rapid o copie de rezervă.

Anexa A: Gestiunea modificărilor documentului

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| 1.0 | 13/04/2025 | Bratu Vlad-Cristian | Crearea documentului de specificații inițiale |
| 1.1 | 14/04/2025 | Bratu Vlad-Cristian | Adăugarea detaliilor pentru arhitectura software |
| 1. | 15/04/20  25 | Bratu Vlad-Cristian | Corectarea secțiunii de performanță și optimizare |

Anexa B: Acronime

Instrucțiuni: Furnizați o listă de acronime și traduceri literale asociate utilizate în cadrul documentului. Enumerați acronimele în ordine alfabetică folosind un format tabular, așa cum este ilustrat mai jos.

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| NLP | Natural Language Processing |
| HTTP | HyperText Transfer protocol |
| API | Aplication Programming Interface |

Anexa C Documente la care se face referire

Instrucțiuni: Sintetizați relația acestui document cu alte documente relevante. Furnizați informații de identificare pentru toate documentele folosite pentru a ajunge la și/sau referite în acest document (de exemplu, documente conexe și/sau asociate, documente prealabile, documentație tehnică relevantă, etc.).

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| Documentul1 | /docs/Document de cerinte (SRS)/Laborator 1 | <ZZ/LL/AAAA> |