<Recunoașterea genurilor de știri folosind inteligența artificială utilizând algoritmi de machine learning | SRCSCUAML>

# Documentul de proiectare

Cuprins

Cuprins

[Documentul de proiectare i](#_Toc196704838)

[Cuprins ii](#_Toc196704839)

[1. Introducere 1](#_Toc196704840)

[1.1 Scopul documentului 1](#_Toc196704841)

[2. Prezentare generală și abordări de proiectare 2](#_Toc196704842)

[2.1 Prezentare generală 2](#_Toc196704843)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc196704844)

[3. Considerații de proiectare 5](#_Toc196704845)

[3.1 Obiective și linii directoare (ghiduri) 5](#_Toc196704846)

[3.2 Metode de dezvoltare 6](#_Toc196704847)

[3.3 Strategii de arhitectură 6](#_Toc196704848)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 7](#_Toc196704849)

[4.1 Vedere logică 7](#_Toc196704850)

[4.2 Arhitectură hardware 7](#_Toc196704851)

[4.3 Arhitectură software 8](#_Toc196704852)

[4.4 Arhitectura informațiilor 8](#_Toc196704853)

[4.5 Arhitectura de comunicații interne 8](#_Toc196704854)

[4.6 Diagrama de arhitectură a sistemului 9](#_Toc196704855)

[5. Proiectarea sistemului 10](#_Toc196704856)

[5.1 Proiectarea bazei de date 10](#_Toc196704857)

[5.2 Conversii de date 11](#_Toc196704858)

[5.3 Interfețe utilizator 11](#_Toc196704859)

[5.4 Proiectarea interfețelor cu utilizatorul 12](#_Toc196704860)

[6. Scenarii de utilizare 14](#_Toc196704861)

[7. Proiectare de detaliu 16](#_Toc196704862)

[7.1 Proiectare hardware de detaliu 16](#_Toc196704863)

[7.2 Proiectare software de detaliu 17](#_Toc196704864)

[7.3 Proiectare detaliată de securitate 19](#_Toc196704865)

[7.4 Proiectare de detaliu pentru performanța sistemului 19](#_Toc196704866)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 20](#_Toc196704867)

## Introducere

În prezent, nu există multe opțiuni pentru identificarea rapidă și precisă a speciilor de ciuperci, în special pentru utilizatorii care nu au o pregătire profesională în botanică, agricultură sau informatică.

Situația propusă: Dezvoltarea unui site web bazat pe Django, integrând o rețea neuronală care utilizează tehnici de învățare profundă (Deep Learning) pentru a identifica și clasifica diferite specii de ciuperci pe baza imaginilor încărcate de utilizatori. Utilizatorii vor putea să încarce fotografii și să primească informații precise despre speciile respective, inclusiv descrierea caracteristicilor, utilizări, și alte informații utile.

### Scopul documentului

Scopul documentului de specificații pentru tema aleasă este de a furniza o descriere detaliată și clară a cerințelor, funcționalităților și constrângerilor proiectului, în acest caz, un site web care utilizează rețele neuronale pentru identificarea speciilor de ciuperci. Acest document servește ca un ghid pentru toate părțile implicate în utilizarea acestui sistem.

Astfel, printre scopuri se regăsesc:

1. Clarificarea cerințelor funcționale și non-funcționale
2. Detaliereea tehnologiilor și arhitecturii sistemului
3. Definirea criteriilor de performanță
4. Ghid pentru testare și validare
5. Securitate și confidențialitate
6. Planificarea și gestionarea resurselor
7. Clarificarea responsabilităților și a fluxului de lucru
8. Determinarea limitărilor și a constrângerilor

Prin urmare, documentul de specificații este esențial pentru a asigura o viziune comună asupra proiectului, pentru a ghida dezvoltarea tehnică și pentru a stabili așteptări clare privind performanța, securitatea și funcționalitatea sistemului.

## Prezentare generală și abordări de proiectare

### Prezentare generală

Sistemul propus are ca scop identificarea automată a speciilor de ciuperci pe baza imaginilor încărcate de utilizatori. Acesta va ajuta utilizatorii să identifice rapid diferite specii de ciuperci, folosind tehnici de învățare profundă, și va fi accesibil printr-un site web dezvoltat cu framework-ul Django. Aplicația va permite utilizatorilor să încarce fotografii ale ciupercilor și să obțină informații relevante despre specia identificată, bazându-se pe o rețea neuronală antrenată pe un set larg de imagini de ciuperci, oferind un punct de vedere asupra toxicității acesteia.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Presupuneri privind utilizatorii: Utilizatorii trebuie să aibă acces la o conexiune stabilă la internet pentru a putea utiliza aplicația web, deoarece funcționarea acesteia depinde de încărcarea și procesarea imaginilor pe server. Totodată, aceștia trebuie să fie capabili să încarce imagini de ciuperci în format digital (JPG, PNG și așa mai departe), presupunând un nivel minim de cunoștințe tehnice pentru încărcarea imaginii și interpretarea rezultatelor oferite de către aplicație.

Presupuneri tehnice: Rețeaua neuronală va fi antrenată corect și va obține un nivel de performanță acceptabil pentru identificarea ciupercilor, cu o acuratețe de clasificare suficientă (precum 80-90% în funcție de complexitatea datelor). Este posibil ca în timp modelul să aibă nevoie de antrenamente suplimentare pentru a îmbunătăți performanța. Timpul de răspuns al sistemului pentru procesarea unei imagini va fi de aproximativ 2-5 secunde pentru majoritatea imaginilor. Cu toate acestea, este posibil ca în funcție de complexitatea rețelei neuronale și dimensiunea imaginilor să existe o ușoară variație a acestui timp.

Presupuneri privind securitatea și confidențialitatea: Aplicația va respecta reglementările privind protecția datelor, cum ar fi GDPR, având în vedere că utilizatorii vor încărca imagini ce pot fi considerate date personale iar utilizatorii vor trebui să accepte o politică de confidențialitate care detaliază modul în care sunt gestionate și stocate datele lor.

#### Constrângeri

Constrângeri legate de precizia modelului de învățare automată:

* Acuratețea identificării: Performanța rețelei neuronale poate varia în funcție de complexitatea imaginii sau de calitatea acesteia (luminozitate, rezoluție, unghiuri diferite ale ciupercilor, etc.). Chiar și un model bine antrenat poate comite erori în cazul unor imagini de calitate scăzută sau în situațiile în care ciupercile seamănă foarte mult între ele
* Lipsa diversității datelor: Dacă rețeaua neuronală nu a fost antrenată pe o gamă largă de imagini cu toate varietățile de ciuperci, va fi mai greu pentru sistem să recunoască corect specii mai puțin comune sau variante mai rare ale unei ciuperci
* Generalizarea modelului: Modelul poate întâmpina dificultăți în identificarea ciupercilor care nu sunt incluse în setul de date folosit pentru antrenament. De exemplu, specii noi sau neobișnuite de ciuperci ar putea fi mai greu de recunoscut.

Constrângeri legate de hardware și performanță:

* Timpul de procesare: Deși procesarea imaginilor poate fi rapidă pentru majoritatea ciupercilor, în funcție de dimensiunea imaginii și complexitatea modelului, timpul de procesare poate crește. Dacă serverul nu este suficient de puternic (în special pentru antrenarea rețelei neuronale), procesarea în timp real poate să nu fie suficient de rapidă.
* Resurse hardware pentru învățarea profundă: Antrenarea unui model de învățare profundă eficient poate necesita resurse hardware avansate, precum unități de procesare grafică (GPU), ceea ce poate adăuga costuri semnificative la infrastructura necesară. În plus, performanța serverului poate scădea pe măsură ce numărul de utilizatori simultani crește.

Constrângeri de utilizabilitate:

* Calitatea imaginii: Calitatea imaginii încărcate este crucială pentru precizia identificării. Utilizatorii care încarcă imagini de proastă calitate (de exemplu, blurate, cu lumină slabă, cu unghiuri neclare) pot obține rezultate mai puțin precise. Aplicarea unui filtru de preprocesare a imaginilor poate ajuta la îmbunătățirea rezultatelor, dar nu va rezolva complet problema.
* Complexitatea interfeței utilizator: Aplicația trebuie să fie simplu de utilizat pentru toate categoriile de utilizatori, indiferent de nivelul lor de cunoștințe tehnice. Complexitatea interfeței sau a procesului de încărcare a imaginilor poate deveni o barieră pentru utilizatori.
* Necesitatea unei conexiuni stabile la internet: Aplicația depinde de o conexiune la internet pentru a funcționa corect, ceea ce ar putea reprezenta o constrângere pentru utilizatorii din zone cu acces limitat la internet.

#### Riscuri

* Utilizatorii ar putea obține rezultate incorecte sau confuze, ceea ce ar afecta încrederea lor în sistem și ar duce la scăderea utilizării aplicației. De asemenea, ar putea fi necesar un efort suplimentar pentru a îmbunătăți modelul prin colectarea de noi date de antrenament și ajustarea arhitecturii rețelei neuronale.
* Aplicația poate deveni lentă sau inaccesibilă în perioadele de vârf, ceea ce ar reduce satisfacția utilizatorilor și ar duce la întreruperi ale serviciului. Acest risc ar necesita implementarea unei infrastructuri scalabile, cum ar fi cloud-ul, pentru a răspunde cerințelor de trafic în creștere.
* Atacurile cibernetice pot compromite integritatea și disponibilitatea aplicației, iar datele utilizatorilor pot fi expuse. Pentru a preveni acest risc, aplicația trebuie să aibă implementate măsuri de securitate robuste, inclusiv criptarea datelor, autentificare securizată și protecție împotriva atacurilor de tip injection.
* Incompatibilitatea cu unele platforme sau browsere poate duce la o experiență de utilizare neplăcută și la pierderea unor utilizatori. Este important să se efectueze teste riguroase pe multiple platforme și să se asigure compatibilitatea aplicației cu toate acestea.
* Aplicația ar putea depinde de servicii externe pentru stocarea datelor sau pentru procesarea imaginii (de exemplu, servicii cloud sau API-uri externe pentru procesarea imaginilor). Dacă aceste servicii externe devin inaccesibile sau înregistrează erori, aplicația ar putea să nu funcționeze corect. Acest risc poate fi atenuat prin monitorizarea constantă a serviciilor externe și prin implementarea unor soluții de backup sau redundanță.
* Modificările necontrolate pot afecta stabilitatea aplicației și pot introduce erori sau incompatibilități cu versiunile anterioare ale sistemului. Este esențial să se adopte un proces riguros de management al versiunilor și să se efectueze teste temeinice înainte de a implementa modificări.

## Considerații de proiectare

Printre problemele ce trebuiesc abordate într-o etapă incipientă se regăsesc: stabilirea utilizatorilor principali (precum botaniștii, agricultorii, pasionații), funcționalitățile pe care trebuie să le aibă platforma, fluxurile principale de utilizare, nivelul minim de acuratețe acceptabil pentru algoritmul de ML, calitatea și diversitatea datelor folosite în antrenarea modelului de Machine Learning, alegerea arhitecturii potrivite pentru model, astfel încât această să satisfacă nevoile utilizatorilor, gestionarea scalării platformei, designul arhitectural (precum separarea componentelor, comunicarea între containere și persistența datelor), securitatea, interacțiunea dintre platformă și utlizator sau testarea și mentenanța.

### Obiective și linii directoare (ghiduri)

Obiective principale:

Precizie și performanță: modelul ML trebuie să fie antrenat pentru a oferi rezultate relevante și precise, însă timpul de inferență trebuie să fie cât mai redus, ideal în timp real sau aproape real

Experiență de utilizare și funcționalitate: interfața trebuie să fie ușor de folosit, inclusiv de către utilizatorii non-tehnici, însă funcționalitățile trebuie să fie diverse, astfel încât sunt acoperite cât mai multe dintre nevoile persoanelor ce folosesc platforma, pentru a nu fi nevoie să apeleze mai multe servicii

Principii și linii directoare de design software:

Separarea clară a componentelor: backend-ul, modelul ML și baza de date trebuie să constituie servicii independente

Modularitate: codul trebuie să fie organizat modular, pentru a permite înlocuirea sau îmbunătățirea componentelor fără a le afecta pe celelalte (spre exemplu îmbunătățirea modelului de ML, fără a afecta celelalte resurse)

Reproductibilitate și consistență prin containerizare: containerele Docker vor avea aceiași configurație indiferent de mediul în care se află (dezvoltare, testare, producție). Totodată, imaginile trebuie să includă aceleași versiuni, indiferent de mediu, pentru asigurarea funcționării optime.

Priorități de securitate și protecția datelor:

Gestionarea sigură a datelor: criptarea și folosirea protocolului HTTPS, respectarea GDPR

Protecție împotriva atacurilor comune: validarea și filtrarea fișierelor încărcate (a imaginilor, în primă instanță), prevenirea atacurilor de tip „injection” (XSS, CSRF, ș.a.m.d)

Logging și monitorizare: loguri clare pentru debuggind și monitorizarea resurselor și a stării containerelor sau a aplicației de-a-ntregul.

### Metode de dezvoltare

Abordare bazată pe arhitectură modulară și orienttă pe servicii (SoA – service oriented architecture / microservicii):

Analiza cerințelor și a cazurilor de utilizare, precum identificarea actorilor și stabilirea funcționalităților cheie

Modularea arhitecturii sistemului, ceea ce presupune separarea platformei în module, definirea interacțiunilor între componente prin REST API

Proiectarea containerelor Docker, astfel încât fiecare componentă rulează într-un container separat, într-o rețea comună

Definirea interfețelor și comunicării între servicii, spre exemplu încărcarea, stocarea și procesarea asincronă a fotografiilor, comunicarea cu modelul ML printr-un endpoint bine stabilit, folosit în întreaga platformă

### Strategii de arhitectură

Tehnologii utilizate:

* Backend & Frontend: Django (Python)
* Web server: nginx
* WSGI: Gunicorn
* Machine Learning: Rețele neuronale
* UI Component Library: DaisyUI
* Baza de date: PostgreSQL, Redis
* Alte tehnologii: TensorFlow pentru dezvoltarea și antrenarea rețelei neuronale, Docker pentru containerizare, Pandas pentru interacțiunea cu fișierele, numpy pentru calcularea eficientă a elementelor necesare antrenării modelului de inteligență artificială, Websocket pentru interacțiunea în timp real între client și server

## Arhitectura Sistemului și Proiectarea Arhitecturii

Arhitectura sistemului este concepută pe baza principiilor de modularitate, scalabilitate și separare clară a responsabilităților, fiind organizată în jurul unui model de tip multi-serviciu containerizat. Sistemul este compus din mai multe componente specializate, fiecare rulând în propriul container Docker, comunicând între ele prin interfețe bine definite (API-uri REST sau baze de date comune). Această arhitectură permite extinderea, întreținerea și scalarea sistemului în mod eficient.

Fluxul de date și interacțiunea între componente constă în:

Interacțiunea utilizatorului cu aplicația: utilizatorul interacționeză cu interfața web, în special pentru a încărca o imagine cu o specie de ciupercă, așteptând un răspuns bazat pe imaginea încărcată, moment în care platforma procesează cererea direct, printr-o rută HTTP și preia fișierul încărcat

Procesarea și delegarea către serviciul ML: imaginea este validată și stocată (temporar sau permanent), ulterior fiind transferată către serviciul ML

Execuția algoritmului de I.A: serviciul de machine learning preia imaginea, rulează clasificarea și trimite un răspuns înapoi către utilizator, conținând rezultatul

Stocarea și livrarea rezultatului: aplicația salvează rezultatul în baza de date, alături de alte metadate necesare și este afișat utilizatorului în interfața web, generată dinamic

### Vedere logică

Aplicația este organizată în jurul mai multor module funcționale. Utilizatorii interacționează cu sistemul prin intermediul interfeței web servite de Django, care gestionează totul, de la autentificare și gestionarea profilului, până la istoricul clasificărilor anterioare. Modulul de încărcare a imaginilor permite trimiterea fișierelor către sistem, iar un modul dedicat trimite aceste imagini către un serviciu separat de inferență bazat pe machine learning. După primirea rezultatului, un alt modul se ocupă cu prezentarea predicției (specia de ciupercă, scorul de încredere) în interfață. Există și un sistem de notificări în timp real, gestionat prin WebSockets și Redis, care informează utilizatorul despre starea procesării. În plus, un modul administrativ permite gestiunea utilizatorilor și a versiunilor modelului ML.

### Arhitectură hardware

Aplicația rulează într-un mediu virtualizat (sau containerizat), de obicei pe o instanță de tip VPS sau într-un mediu cloud orchestrat, unde resursele hardware sunt bine definite. Sistemul de operare utilizat este de regulă o distribuție Linux (precum Ubuntu), iar configurația hardware poate include un procesor multi-core, între 16 și 64 GB RAM, stocare SSD și un GPU pentru accelerarea inferenței modelului ML. Componentele software sunt distribuite în containere Docker separate, fiecare mapat pe porturi specifice. De exemplu, baza de date PostgreSQL rulează în mod standard pe portul 5432, iar accesul la acest port este restricționat doar la rețeaua internă definită în Docker, prevenind expunerea sa în exterior. Redis folosește portul 6379, configurat similar, cu acces limitat strict la containerele din rețeaua aplicației. Django rulează într-un container separat, pe portul 8000, dar acest port este „în spatele” unui server Nginx care rulează pe portul 80 (pentru HTTP) și 443 (pentru HTTPS). Nginx acționează ca reverse proxy, folosindu-se și de un certificat SSL, ceea ce înseamnă că traficul extern este criptat și securizat. Certificatul SSL este gestionat prin Let's Encrypt sau o autoritate similară. Comunicarea între containere este izolată într-o rețea Docker dedicată, protejată de reguli firewall și politici de acces explicite, asigurând că nicio componentă nu este expusă în afara celor necesare public (de exemplu, doar portul 443).

### Arhitectură software

Din punct de vedere software, aplicația este structurată modular și distribuită în containere Docker care rulează împreună sub orchestrarea „docker-compose”. Django reprezintă nucleul sistemului, gestionând interfața web, logica aplicației și toate rutele HTTP sau WebSocket, fiind rulat sub un server compatibil WSGI. Serviciul ML este dezvoltat independent, ca microserviciu REST, care expune o interfață de inferență și primește imagini pentru clasificare, returnând predicții în format JSON. Redis este utilizat atât pentru sistemul de mesaje al WebSocket-urilor prin Django Channels, cât și pentru caching temporar sau implementarea unui mecanism de job queue asincron. PostgreSQL servește ca sistem principal de gestiune a datelor, cu acces securizat doar din rețeaua internă. Nginx se ocupă de rutarea traficului extern și gestionează securitatea TLS, oferind și o protecție suplimentară împotriva accesului neautorizat prin configurări stricte de headers și rate limiting.

### Arhitectura informațiilor

Structura datelor din aplicație este centrată în jurul câtorva entități principale, printre care utilizatorul, imaginea încărcată și rezultatul inferenței. Fiecare utilizator poate încărca imagini, care sunt stocate împreună cu metadatele necesare procesării, cum ar fi timestamp-ul, formatul și locația fișierului. Fiecare imagine are asociat un rezultat ML, compus din specia prezisă, un scor de încredere și eventuale sugestii alternative. Dacă sistemul gestionează mai multe versiuni ale modelului ML, și acestea pot fi înregistrate ca entități separate. Fluxul de date este bine delimitat: imaginea este preluată de Django, validată și stocată local sau într-un bucket compatibil cu S3. După această etapă, este transmisă către serviciul ML, care returnează predicția. Pe toată durata acestui proces, Redis facilitează comunicarea internă pentru actualizări în timp real, iar după obținerea rezultatului, acesta este salvat în PostgreSQL și transmis înapoi către interfața utilizatorului printr-un canal WebSocket activ.

### Arhitectura de comunicații interne

Comunicarea dintre componentele interne ale aplicației se face printr-un set de protocoale și canale bine definite, toate rulând în cadrul unei rețele Docker izolate. Django trimite cereri HTTP către serviciul ML pentru a obține predicții, folosind interfețe REST simple. În paralel, folosește Redis pentru a publica notificări legate de starea procesării – de exemplu, începutul și finalul inferenței. WebSocket-urile sunt implementate folosind Django Channels, care permite gestionarea în timp real a conexiunilor active ale utilizatorilor. Aceste canale sunt alimentate de mesaje interne prin Redis și pot trimite notificări live către browser imediat ce procesarea unei imagini este completă. Accesul la Redis este restricționat la containerele autorizate, iar traficul între containere nu părăsește rețeaua internă. Comunicarea cu baza de date PostgreSQL se face prin SQL standard, cu autentificare la nivel de utilizator și criptare TLS activată dacă este necesar. Pentru stocarea imaginilor, Django poate folosi fie sistemul local de fișiere, fie un serviciu cloud compatibil S3, cu autentificare pe bază de token și drepturi de acces restrictive. Toate componentele sunt izolate și protejate prin firewall intern, iar serviciile expuse public sunt atent limitate la strictul necesar – în special portul 443 pentru HTTPS, oferind un mediu sigur și controlat pentru întreaga aplicație.

## Proiectarea sistemului

### Proiectarea bazei de date

Proiectarea bazei de date presupune crearea unei structuri care să sprijine funcționalitățile platformei, să stocheze eficient datele și să asigure integritatea și performanța sistemului. În acest caz, baza de date va fi gestionată de PostgreSQL și va include mai multe tabele principale. Tabelul „users” va stoca informațiile despre utilizatori, incluzând id-ul, numele, email-ul și parola criptată, precum și rolurile acestora. Tabelul „images” va stoca informațiile referitoare la imaginile încărcate de utilizatori, cum ar fi id-ul, utilizatorul asociat, numele fișierului și data încărcării. Tabelul „inferences” va salva predicțiile făcute de modelul de ML, incluzând specia prezisă, scorul de încredere și data predicției. Tabelul „models” va stoca versiunile modelului de ML, iar tabelul „logs” va păstra logurile detaliate ale activității utilizatorilor și ale sistemului. Aceste tabele vor fi legate între ele prin chei primare și străine pentru a asigura integritatea referențială.

#### Obiecte de date și structuri de date rezultante

Obiectele de date și structurile de date rezultate în cadrul platformei vor reflecta atât entitățile din baza de date, cât și entitățile care sunt procesate la nivelul aplicației. Obiectul „User” va reprezenta un utilizator al platformei și va conține informații precum id-ul, numele, email-ul și rolurile. Acesta va fi legat la tabelul „users” din baza de date. Obiectul „Image” va reprezenta o imagine încărcată de utilizator și va include id-ul, numele fișierului, referința la utilizatorul care a încărcat-o și data încărcării, fiind legat la tabelul „images”. Obiectul „Inference” va reprezenta o predicție efectuată de modelul ML, incluzând specia prezisă, scorul de încredere și timestamp-ul predicției. Acesta va fi legat la tabelul „inferences”. De asemenea, obiectul „MLModel” va reprezenta un model de învățare automată, incluzând versiunea și datele de instruire, fiind legat la tabelul „models”. Structurile de date, precum listele, dicționarele și tuplurile, vor fi utilizate pentru manipularea datelor temporare, de exemplu pentru a stoca parametrii unei predicții ML într-un dicționar.

#### Fișiere și baze de date

Fișierele și bazele de date vor juca un rol esențial în platformă. Fișierele de imagini vor fi încărcate de utilizatori și stocate pe un server local sau într-un serviciu extern de tip S3 (Amazon S3). Informațiile despre aceste fișiere (de exemplu, calea către fișierul respectiv) vor fi salvate în tabela „images” a bazei de date. Fișierele de loguri vor înregistra activitățile sistemului și ale utilizatorilor și vor fi stocate pe server, în directorul „/logs”. Aceste fișiere vor fi accesibile doar administratorilor pentru a monitoriza starea platformei și pentru debugging. Baza de date PostgreSQL va fi folosită pentru stocarea datelor tranzacționale, cum ar fi informațiile despre utilizatori, imaginile încărcate și rezultatele predicțiilor.

##### Baze de date

Proiectarea detaliată a fișierelor DBSM (Database Management System) pentru PostgreSQL include mai multe fișiere esențiale. Fișierele de configurare ale PostgreSQL, cum ar fi fișierele care stabilesc setările de conectare, performanța și securitatea, sunt de obicei stocate în directorul /etc/postgresql/. Fișierele de date, care conțin efectiv informațiile tabelelor, sunt gestionate de motorul de baze de date și sunt stocate într-un director specific, de obicei în /var/lib/postgresql/data/. De asemenea, PostgreSQL utilizează fișiere de jurnal (Write-Ahead Log - WAL) pentru a asigura integritatea tranzacțiilor și pentru a sprijini recuperarea bazei de date în caz de eșec. Aceste fișiere sunt stocate într-un director separat.

##### Fișiere non-DBMS

Fișierele non-DBSM sunt fișierele care nu sunt gestionate direct de sistemul de baze de date, dar care sunt necesare pentru funcționarea platformei. Fișierele de imagini sunt încărcate de utilizatori și sunt stocate pe servere externe sau în sistemele de stocare în cloud (precum S3). Deși aceste fișiere nu sunt gestionate de PostgreSQL, informațiile legate de ele (de exemplu, calea fișierului) sunt salvate în tabelul „images”. Fișierele de configurare, cum ar fi requirements.py, settings.py din cadrul aplicației Django sau docker-compose.yml din mediul Docker, sunt necesare pentru a seta corect comportamentul aplicației și nu sunt stocate în baza de date. De asemenea, fișierele de loguri vor fi stocate în fișiere externe pentru a înregistra activitățile sistemului, fără a fi stocate direct în baza de date.

### Conversii de date

Conversiile de date sunt esențiale pentru a asigura interoperabilitatea între diferitele componente ale platformei. De exemplu, atunci când un utilizator încarcă o imagine, aceasta poate fi convertită într-un format standardizat (de exemplu, JPEG sau PNG) înainte de a fi stocată sau procesată de modelul de ML. Conversia între formatele de date JSON și SQL va fi folosită atunci când aplicația Django trimite și primește date între API și baza de date. De asemenea, datele generate de modelul de ML, cum ar fi predicțiile (specia prezisă și scorul de încredere), vor fi convertite într-un format ușor de procesat și prezentat utilizatorului, de obicei sub formă de JSON, la fel ca și celelalte răspunsuri oferite de către platformă prin intermediul rutelor de tip „Open API”.

### Interfețe utilizator

În cadrul acestei platforme web, există mai multe clase de utilizatori sau roluri asociate, fiecare având un set specific de permisiuni și responsabilități. Rolul de „Utilizator” (standard) este rolul de bază atribuit utilizatorilor care se înregistrează pe platformă. Utilizatorii standard pot încărca imagini ale ciupercilor, pot vizualiza predicțiile făcute de modelul ML, având la dispoziție istoricul acestora într-un profil personal. Aceștia nu au permisiuni administrative și nu pot modifica setările platformei. Rolul de „Administrator” are acces complet la platformă. Administratorii pot gestiona utilizatorii, vizualiza toate imaginile și predicțiile, pot modifica setările aplicației și pot gestiona modelele de învățare automată utilizate în sistem. Administratorii pot accesa logurile complete ale activității utilizatorilor și ale sistemului, având și responsabilitatea de a efectua mentenanța platformei. Operatorii ML sunt responsabili pentru gestionarea și întreținerea modelelor de învățare automată. Ei pot încărca noi versiuni ale modelului de ML, pot monitoriza performanța acestuia și pot analiza erorile legate de predicții, însă nu au acces la datele utilizatorilor.

#### Intrări

Utilizatorii și operatorii vor interacționa cu platforma prin mai multe mijloace de intrare. Cel mai important mijloc va fi interfața web, disponibilă printr-un browser, care le va permite utilizatorilor să încarce imagini ale ciupercilor, să vizualizeze rezultatele predicțiilor și să salveze istoricul acestora. De asemenea, utilizatorii vor putea crea conturi, modifica datele personale și gestiona setările de profil. Utilizatorii vor utiliza formulare de upload pentru a încărca imagini personale ale ciupercilor în scopul identificării acestora. Aceste formulare vor permite utilizatorilor să selecteze fișiere de pe dispozitivele lor sau din servicii externe de stocare, precum Google Drive sau Dropbox. Interfețele de configurare vor fi utilizate de către administratori pentru a ajusta setările aplicației, inclusiv parametrii de securitate, performanță și gestionarea utilizatorilor. Aceste interfețe vor include formulare pentru configurare, tabele pentru gestionarea utilizatorilor și a permisiunilor, precum și setări de server.

#### Ieșiri

Sistemul va produce mai multe tipuri de ieșiri, care vor fi prezentate utilizatorilor și administratorilor în funcție de rolurile lor. Un exemplu de ieșire sunt rapoartele de predicție, care vor fi generate după ce un utilizator încarcă o imagine pentru identificarea unei ciuperci. Aceste rapoarte vor include specia prezisă, scorul de încredere, și posibilele sugestii suplimentare pentru confirmarea speciei, iar utilizatorii vor putea vizualiza aceste rezultate pe ecranul principal. Interfețele grafice de monitorizare vor fi accesibile administratorilor și vor permite vizualizarea statisticilor despre performanța modelului, numărul de imagini procesate, precizia și erorile întâlnite. Aceștia vor putea vizualiza grafice de performanță și vor avea opțiunea de a răspunde la anomaliile apărute. Paginile aferente vizualizării istoricului de predicții vor permite utilizatorilor să vizualizeze istoricul predicțiilor efectuate pe contul lor. Aceste pagini vor afișa o listă cu imaginile și rezultatele corespunzătoare, incluzând specia prezisă și scorul de încredere, oferind de asemenea opțiuni pentru a căuta și filtra predicțiile anterioare. Totodată, erorile și mesajele de log vor fi accesibile administratorilor pentru diagnosticarea problemelor apărute în platformă. Aceste mesaje pot include erori de server, mesaje de diagnostic ale aplicației și alte informații utile. Fișierele de ieșire, care pot fi de tip CSV sau JSON, vor fi generate pentru a permite utilizatorilor și administratorilor să salveze rezultatele și să le utilizeze pentru analize suplimentare sau arhivare.

### Proiectarea interfețelor cu utilizatorul

Interfețele platformei sunt cruciale pentru interacțiunea utilizatorilor și administratorilor cu sistemul. Prima dintre acestea este interfața de autentificare și cea de înregistrare, care permite utilizatorilor să se înregistreze sau să se autentifice folosind un email și o parolă. Utilizatorii vor putea, de asemenea, să recupereze parolele uitate sau să își modifice contul. A doua interfață importantă este interfața de încărcare a imaginilor, care va permite utilizatorilor să încarce fișiere de pe dispozitivele lor sau din serviciile de stocare externe. Interfața va include un buton de selectare a fișierelor, o previzualizare a imaginii încărcate și opțiuni pentru a trimite imaginea spre identificare. După procesarea imaginii, utilizatorii vor putea vizualiza rezultatele într-o interfață de vizualizare a predicțiilor, care va arăta specia prezisă, scorul de încredere și alte informații relevante. De asemenea, utilizatorii vor putea salva sau partaja aceste rezultate. Interfața de administrare, destinată administratorilor, le va permite acestora să gestioneze utilizatorii, să vizualizeze logurile și să modifice setările platformei. Aceștia vor putea adăuga sau șterge utilizatori, actualiza modelele de ML și ajusta configurațiile platformei. De asemenea, interfața de monitorizare a performanței permite urmărirea performanța modelelor de machine learning. În aceiași pagină vor putea fi vizualizate statistici de performanță, rapoarte de erori și vor putea fi încărcate noi versiuni ale modelului pentru a îmbunătăți rezultatele predicțiilor.

## Scenarii de utilizare

Printre scenariile de utilizare, se regăsesc:

Un utilizator nou accesează platforma și vizualizează pagina principală. Alege opțiunea de a se înregistra pe platformă și completează un formular cu informațiile sale personale, cum ar fi numele, adresa de email și parola. După înregistrare, utilizatorul se autentifică pe platformă folosind emailul și parola create. Odată autentificat, utilizatorul ajunge pe pagina principală unde poate încărca o imagine a unei ciuperci pentru a o identifica. Apasă pe butonul „Încărcați imaginea” și selectează fișierul din dispozitivul său. După ce fișierul este încărcat, platforma procesează imaginea și în câteva secunde afișează rezultatele predicției, incluzând specia prezisă, scorul de încredere și o descriere detaliată. Utilizatorul poate salva rezultatul și poate vizualiza predicțiile anterioare în secțiunea „Istoric”.

Un administrator se autentifică pe platformă folosind contul său de administrator. Ajunge pe panoul de control, unde vizualizează un tabel cu toți utilizatorii platformei, incluzând informații precum numele, emailul și statutul lor (activ/inactiv). Administratorul decide să dezactiveze un utilizator care a încălcat termenii de utilizare ai platformei. Apasă pe opțiunea „Dezactivează” din tabelul de administrare, iar utilizatorul respectiv va fi blocat de la utilizarea platformei. De asemenea, administratorul poate adăuga un nou utilizator sau poate modifica permisiunile utilizatorilor existenți, dacă este necesar. Administratorul poate vizualiza logurile activității utilizatorilor și poate identifica erori sau comportamente suspecte ale acestora.

Totodată, un administrator poate accesa secțiunea de gestionare a modelelor de învățare automată. Acesta vede că performanța actualului model a scăzut, iar sistemul nu mai identifică corect anumite specii de ciuperci. Astfel, administratorul poate decide să-l înlocuiască pe cel vechi. Încarcă fișierul noului model pe platformă printr-un formular de încărcare, iar platforma actualizează automat configurațiile pentru a utiliza noul model. După încărcare, pot fi vizualizate statistici despre performanța noului model, verificându-se dacă îmbunătățirile au fost benefice. Se pot analiza erorile întâmpinate și se pot ajusta parametrii modelului pentru a obține rezultate mai bune.

Un utilizator care a mai folosit platforma anterior dorește să verifice istoricul predicțiilor efectuate pe parcursul utilizării platformei. Se conectează la contul său și accesează secțiunea „Istoric” din meniul principal. Aici, utilizatorul vizualizează toate imaginile încărcate anterior, împreună cu rezultatele predicțiilor (specia prezisă, scorul de încredere și o descriere a speciei de ciupercă). Utilizatorul poate filtra rezultatele după data predicției sau poate căuta o anumită ciupercă folosind funcția de căutare. Dacă utilizatorul dorește să păstreze informațiile pentru referințe ulterioare, poate exporta istoricul predicțiilor într-un fișier CSV sau JSON.

De asemenea, un utilizator poate raporta faptul că platforma se deschide prea greu și că întâmpină întârzieri în momentul în care încarcă o imagine pentru identificarea unei ciuperci. În acest caz, administratorul se va autentifica pe platformă și va accesa panoul de administrare, unde poate vizualiza logurile aplicației și ale serverului. În cazul în care observă o întârziere semnificativă la procesarea imaginilor, se va decide să verifice și setările serverului, inclusiv resursele de procesare alocate. După identificarea unor erori în configurația serverului, vor fi ajustate setările și optimizată performanța serverului. După efectuarea ajustărilor, platforma devine mai rapidă, iar utilizatorii nu mai întâmpină întârzieri.

## Proiectare de detaliu

Pentru construirea și integrarea efectivă a componentelor hardware și software într-un sistem funcțional, echipa de dezvoltare trebuie să urmeze pași detaliați și bine structurați. Serverul trebuie să fie conectat la rețele interne de tip local (LAN) și externe (Internet), cu porturi deschise pentru a permite comunicațiile cu bazele de date, API-uri externe și modulele de ML. Configurarea serviciilor externe de stocare sau platformele de cloud (AWS, Google Cloud, Azure, dacă este cazul) pentru procesarea și stocarea datelor la scară largă. Pentru comunicarea eficientă între servere și platformă, este necesară configurarea unui serviciu de Redis pentru gestionarea sesiunilor și al mesajelor în timp real (ex: pentru actualizările de stare ale identificării speciilor de ciuperci sau pentru gestionarea cererilor concurente). Arhitectura aplicației va fi construită pe baza framework-ului Django, care va gestiona toate cerințele API-urilor RESTful și va interacționa cu baza de date PostgreSQL. Django va asigura și securitatea aplicației prin autentificare și autorizare. Se va utiliza Docker pentru a izola și containeriza aplicațiile și serviciile, facilitând implementarea pe diferite medii (precum dezvoltare, testare sau producție). Fiecare componentă va fi containerizată în propriul său container. Fluxul de date între componentele hardware și software este esențial pentru buna funcționare a platformei. Aplicația va rula pe serverul hardware configurat, unde va interacționa cu baza de date (PostgreSQL) și va comunica cu modelul de învățare automată stocat pe servere dedicate sau în cloud. La încărcarea unei imagini, utilizatorul va trimite fișierul către backend-ul Django, care va trimite imaginea procesată la modelul ML (prin API-ul creat). Modelul ML va analiza imaginea și va returna rezultatul la backend-ul Django, care îl va oferi înapoi utilizatorului un răspuns.

### Proiectare hardware de detaliu

Serverul principal este componenta centrală a infrastructurii hardware. Este recomandat să se utilizeze servere dedicate pentru a găzdui atât aplicația web (bazată pe Django), cât și bazele de date (PostgreSQL) și modelele de învățare automată. Serverele ar trebui să fie echipate cu procesoare multi-core performante, precum Intel Xeon sau AMD EPYC, care pot gestiona cerințele mari de procesare și de paralelism ale aplicației și ale algoritmilor de învățare automată. Aceste servere vor trebui să aibă cel puțin 16 GB de memorie RAM, de preferat 32 GB sau mai mult, pentru a asigura o performanță optimă în prelucrarea datelor voluminoase, în special pentru imagini și modelele de machine learning. În ceea ce privește stocarea, serverele vor necesita SSD-uri rapide cu o capacitate de cel puțin 128 GB sau mai mult, pentru a stoca eficient bazele de date și modelele antrenate de învățare automată. Este important ca SSD-urile să aibă un nivel ridicat de IOPS (Input/Output Operations Per Second) pentru a sprijini operațiunile de citire și scriere rapide. De asemenea, este recomandat să se implementeze soluții de backup automate și redundante, cum ar fi un sistem RAID (Redundant Array of Independent Disks), pentru a proteja datele critice ale utilizatorilor și modelele de învățare automată.

Având în vedere faptul că sistemul utilizează algoritmi de învățare automată, utilizarea plăcilor grafice (GPU) este esențială pentru accelerarea procesării și antrenării modelelor de machine learning. Plăcile grafice NVIDIA, cum ar fi seria Tesla sau RTX sunt recomandate datorită performanței ridicate în sarcini de paralelism masiv, care sunt esențiale pentru antrenarea rapidă a modelelor de învățare profundă (deep learning). Aceste plăci grafice pot sprijini procesele de inferență a modelului de învățare automată, permițând procesarea rapidă a unui număr mare de imagini pentru identificarea speciilor de ciuperci.

Infrastructura hardware va include unul mai multe servere și echipamente de stocare, prin urmare este important să se investească într-un sistem de alimentare fiabil. Acesta ar trebui să includă surse de alimentare neîntreruptibilă (UPS) pentru a proteja echipamentele hardware de fluctuațiile de tensiune sau întreruperile de curent. De asemenea, este necesar să se implementeze un sistem de răcire adecvat pentru servere, având în vedere că acestea vor lucra continuu sub o încărcătură mare, mai ales atunci când sunt folosite GPU-urile pentru procesare.

În funcție de volumul de date și de nevoia de scalabilitate, poate fi necesar să se adauge soluții de stocare în cloud sau externe pentru a sprijini stocarea și procesarea în masă a datelor. Soluțiile de cloud precum AWS, Google Cloud sau Microsoft Azure pot oferi capacități de stocare extinse, precum și putere de calcul suplimentară în momentul în care aplicația se află în fața unor cerințe de performanță ridicate. Aceste soluții pot fi integrate ușor în arhitectura de servere locale, facilitând gestionarea datelor și a resurselor de calcul distribuite.

Securitatea hardware joacă un rol esențial în protejarea platformei. Este necesar să se implementeze măsuri de securitate la nivelul rețelelor interne și externe. Aceste măsuri ar trebui să includă protecție împotriva atacurilor de tip DDoS (Distributed Denial of Service), criptarea datelor transmise între servere și între utilizatori, și protecția datelor sensibile stocate pe servere prin utilizarea tehnologiilor de criptare de tip AES sau SSL. De asemenea, accesul fizic la echipamentele hardware trebuie să fie restricționat și controlat pentru a preveni accesul neautorizat.

### Proiectare software de detaliu

Backend-ul aplicației va fi realizat folosind framework-ul Django, care va gestiona logica aplicației și va oferi interfața de interacțiune cu utilizatorul. Django va fi responsabil pentru gestionarea autentificării utilizatorilor, autorizației, gestionarea sesiunilor și procesarea cererilor API. De asemenea, va interacționa cu baza de date PostgreSQL, stocând și extrăgând informațiile relevante despre utilizatori și specii de ciuperci. Un serviciu important va fi API-ul RESTful care va permite aplicației să accepte cereri de identificare a ciupercilor și să returneze rezultatele procesării, în format JSON. Acest serviciu va trebui să proceseze imagini de la utilizatori, să le trimită la algoritmul de învățare automată pentru a le analiza și să trimită rezultatele înapoi utilizatorului. Django va gestiona, de asemenea, logica de administrare a speciilor de ciuperci, adăugând unele noi sau gestionând informațiile existente.

Unul dintre componentele critice ale aplicației va fi modelul de învățare automată, care va fi responsabil pentru identificarea speciilor de ciuperci pe baza imaginilor încărcate de utilizatori. Acest model este construit folosind un framework de machine learning precum TensorFlow. Inițial, modelul va fi antrenat cu un set de date de imagini de ciuperci etichetate pentru a învăța să recunoască diferite specii. După antrenare, modelul va fi salvat într-un format care poate fi încărcat și utilizat de aplicația Django. Atunci când utilizatorul trimite o imagine pentru procesare, backend-ul va trimite imaginea la modelul ML, care va returna predicția pentru specie.

Baza de date relațională PostgreSQL va fi utilizată pentru a stoca toate datele aplicației, inclusiv informațiile utilizatorilor, datele despre ciuperci și predicțiile făcute de modelul de ML. Baza de date va trebui să fie proiectată eficient, cu tabele și relații care permit stocarea rapidă și accesul ușor la date. Tabelele principale vor include utilizatori, specii de ciuperci, imagini încărcate, predicții și scoruri de încredere. Baza de date va trebui să permită interogări rapide, să gestioneze fișiere mari (imaginile ciupercilor) și să ofere integritate referențială între tabele. De asemenea, va trebui să fie configurată pentru a permite backupuri automate și pentru a asigura securitatea datelor sensibile (spre exemplu criptarea parolelor utilizatorilor).

Aplicația va include un serviciu de evenimente transmise în timp real utilizând WebSockets și Redis. Redis va fi folosit pentru a gestiona sesiuni și a transmite mesaje între servere și clienți în timp real. WebSockets va permite utilizatorilor să primească actualizări în timp real cu privire la progresul procesului de identificare a ciupercilor. De exemplu, utilizatorul va putea să vizualizeze starea procesării imaginii încărcate (exemplu: „Procesare în curs”, „Identificare finalizată"). Acest serviciu va necesita o configurare corectă a canalelor Redis și a conexiunilor WebSocket pentru a asigura o comunicare rapidă și fiabilă. Redis va fi utilizat pentru a stoca și a livra mesaje de la server la client, iar WebSockets va oferi o conexiune bidirecțională între client și server pentru actualizări continue.

Toate componentele software ale aplicației (Django, nGINX, Redis, PostgreSQL) vor fi containerizate folosind Docker. Acest lucru va permite rularea aplicației în medii izolate, ușor de replicat și scalabile. Fiecare componentă va rula într-un container propriu, iar toate aceste containere vor fi orchestrate prin Docker Compose. Astfel, se va asigura o separare clară a responsabilităților între componentele aplicației și va fi ușor să se implementeze, testeze și scaleze serviciile.

Platforma va include un API RESTful care va permite comunicarea dintre frontend și backend. API-ul va gestiona cererile pentru identificarea tipurilor de ciuperci, crearea conturilor de utilizator, autentificarea și autorizarea, și alte acțiuni administrative. Acesta va expune diferite endpoint-uri pentru a permite utilizatorilor să încarce imagini, să obțină predicții sau să acceseze informații despre ciuperci, inclusiv implementarea rutelor „Open API” pentru dezvoltatori.

Testarea sistemului va include teste unitare și teste de integrare pentru fiecare serviciu software. Django va fi echipat cu un framework de testare care va permite testarea funcționalităților API-ului și a logicii aplicației. Testele vor verifica corectitudinea predicțiilor realizate de modelul ML și vor valida performanța acestuia în diverse condiții de utilizare. De asemenea, vor fi efectuate teste de securitate pentru a detecta eventualele vulnerabilități în aplicație.

### Proiectare detaliată de securitate

Securitatea aplicației reprezintă un aspect esențial al proiectării sistemului, având în vedere natura datelor prelucrate și riscurile asociate cu utilizarea internetului. Primul pas important în asigurarea securității va fi implementarea autentificării utilizatorilor. Django utilizează un sistem de autentificare bazat pe parole criptate. Parolele vor fi stocate într-o formă criptată folosind algoritmi precum bcrypt sau Argon2 pentru a preveni atacurile de tip brute-force sau de tip dictionary. De asemenea, aplicația va include un sistem de gestionare a permisiunilor bazat pe roluri, astfel încât fiecare utilizator să poată accesa doar acele funcționalități pentru care are permisiuni explicite, protejând astfel datele sensibile.

Pentru protecția datelor „în tranzit”, toate comunicările între client și server vor fi criptate folosind protocolul HTTPS (SSL/TLS), asigurându-se astfel că informațiile nu pot fi interceptate sau modificate în timpul transmiterii. De asemenea, pentru a preveni atacurile de tip Cross-Site Scripting (XSS) și Cross-Site Request Forgery (CSRF), vor fi implementate filtre de validare a intrărilor și măsuri de protecție adecvate în cadrul aplicației Django, care va verifica toate datele transmise de utilizatori înainte de a le procesa. Atacurile de tip SQL Injection vor fi prevenite prin utilizarea de interogări parametrizate și ORM-ul (Object-Relational Mapping) al Django, care asigură validarea corectă a datelor utilizatorilor. În plus, vor fi implementate monitorizări de securitate pentru a detecta orice activitate neobișnuită pe platformă și pentru a preveni breșele de securitate.

În ceea ce privește protecția împotriva atacurilor DDoS (Distributed Denial of Service), vor fi implementate măsuri de limitare a numărului de cereri per secundă de la aceleași adrese IP și utilizarea unui serviciu de firewall pentru a monitoriza traficul și a bloca orice tentativă de atac. Datele sensibile, cum ar fi informațiile de profil ale utilizatorilor și rezultatele predicțiilor, vor fi criptate și stocate într-un mod sigur în baza de date, iar accesul la aceste informații va fi restricționat prin permisiuni detaliate.

### Proiectare de detaliu pentru performanța sistemului

Performanța aplicației este crucială pentru asigurarea unei experiențe implecabile. Unul dintre aspectele cheie ale performanței va fi optimizarea procesului de inferență al modelului de învățare automată. Acesta va trebui să răspundă rapid la cererile de identificare a ciupercilor încărcate de utilizatori, iar timpul de răspuns al aplicației nu trebuie să depășească câteva secunde. Pentru a îmbunătăți acest timp de răspuns, modelele de învățare automată vor fi optimizate și implementate folosind tehnici de inferență eficientă, care includ reducerea dimensiunii modelelor sau folosirea unor tehnici precum cuantificarea și pruning-ul rețelelor neuronale pentru a le face mai rapide și mai ușor de rulat.

Un alt factor important pentru performanță va fi scalabilitatea sistemului. Aplicarea containerizării cu Docker va permite să se adauge mai multe instanțe ale aplicației în funcție de cerințele de încărcare. În cazul în care volumul de cereri crește, sistemul va putea să scaleze dinamic atât serverele de aplicație, cât și serverele de baze de date, astfel încât să răspundă cerințelor. De asemenea, procesul de gestionare a datelor va fi optimizat, folosind indecși în baza de date pentru a accelera interogările frecvente și pentru a asigura acces rapid la datele necesare.

Pentru a sprijini procesarea rapidă a imaginilor de către modelul ML, aplicația va utiliza plăci grafice dedicate (GPU), care sunt mult mai eficiente în comparație cu procesoarele tradiționale (CPU) pentru sarcini de învățare profundă. Astfel, aplicația va putea să proceseze imagini de mari dimensiuni și să efectueze inferențe în timp real. Serverele de aplicație vor fi echipate cu suficiente resurse de calcul (CPU, RAM, GPU) pentru a răspunde în mod eficient cerințelor sistemului, iar soluțiile de stocare vor fi alese astfel încât să permită citirea și scrierea rapidă a fișierelor de imagini și a datelor utilizatorilor.

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

Comunicarea între componentele sistemului este esențială pentru asigurarea unui flux de lucru eficient. Având în vedere arhitectura bazată pe microservicii și containerizarea cu Docker, fiecare componentă a aplicației va comunica prin intermediul unor canale de comunicare bine definite. Docker joacă un rol esențial în gestionarea acestor comunicări, iar containerizarea va asigura izolare între serviciile care rulează în containere diferite. Containerele vor fi configurate pentru a comunica între ele folosind rețele Docker dedicate. De exemplu, comunicarea între containere se va face prin intermediul rețelelor interne Docker, folosind adrese de tip „docker.host.internal”, care vor permite aplicației să comunice între containere într-un mod rapid și sigur.

În ceea ce privește porturile, aplicația va trebui să aibă porturi bine definite pentru fiecare serviciu, iar aceste porturi vor fi configurate corespunzător în fișierele de configurare Docker (de exemplu, „docker-compose.yml”). Serverul Django va „asculta” pe un port specific (de obicei 8000), în timp ce serverul PostgreSQL va „asculta” pe portul 5432, port standard pentru acest tip de bază de date. În plus, pentru serviciile WebSocket, portul 6379 va fi folosit pentru a comunica cu Redis, care va gestiona sesiuni și mesaje în timp real. De asemenea, serverul Redis va fi configurat să ruleze pe un port dedicat și va gestiona fluxurile de mesaje în cadrul aplicației, inclusiv actualizările în timp real ale statusului procesului de identificare a speciilor de ciuperci.

Pentru a garanta securitatea comunicațiilor, toate porturile de rețea vor fi protejate de firewall-uri și controale de acces, pentru a preveni accesul neautorizat. Comunicarea între componentele interne va fi criptată, iar accesul la serviciile de baze de date va fi restricționat prin autentificare sigură. Utilizarea unui serviciu de orchestration precum Docker Compose va permite echipei de dezvoltare să gestioneze eficient aceste porturi și comunicațiile interne, asigurându-se că toate componentele sunt izolate și că schimbul de date se face într-un mod controlat și securizat.