<Sistem informatic de vizualizare si analiza a datelor geo-spatiale din Romania>

# 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

Acest document descrie specificațiile de proiectare ale unui sistem informatic destinat vizualizării și analizei datelor socio-demografice din România. Sistemul propus urmărește transformarea datelor brute provenite din surse oficiale (precum Institutul Național de Statistică) în informații vizuale relevante și ușor de interpretat prin intermediul unei platforme web interactive.

Documentul de față reflectă modul în care cerințele funcționale și non-funcționale definite anterior sunt transpuse într-o arhitectură tehnică, având ca scop ghidarea procesului de dezvoltare software. Se oferă atât o viziune de ansamblu asupra sistemului, cât și detalii privind proiectarea componentelor software și a interfețelor.

Sistemul va include componente de interacțiune avansată, precum filtre dinamice, hărți tematice și grafice interactive, adaptate nevoilor utilizatorilor finali. Confidențialitatea și integritatea datelor vor fi tratate cu prioritate, prin mecanisme specifice de securitate, iar documentul va evolua odată cu progresul proiectului, reflectând modificările survenite pe parcursul dezvoltării.

### Scopul documentului

Scopul acestui document este de a descrie arhitectura și designul sistemului informatic pentru analiza și vizualizarea datelor socio-demografice din România. Documentul oferă îndrumare tehnică echipei de dezvoltare și sprijină în definirea clară a componentelor sistemului.

Acesta este realizat incremental, fiind adaptat specificului proiectului și metodologiei de dezvoltare utilizate. Publicul țintă include echipa de dezvoltare, managerul de proiect și alte părți interesate, cum ar fi beneficiarii aplicației, în special în secțiunile care privesc interfața cu utilizatorul.

## Prezentare generală și abordări de proiectare

### Prezentare generală

Sistemul informatic propus are ca scop colectarea, prelucrarea, analizarea și vizualizarea datelor socio-demografice din România, într-o manieră accesibilă, interactivă și ușor de interpretat. Platforma va permite utilizatorilor să exploreze datele printr-un dashboard prietenos, dotat cu componente vizuale dinamice: grafice comparative, hărți tematice și filtre contextuale.

Arhitectura aleasă este de tip client-server, bazată pe o aplicație web modulară. Partea client va include interfața grafică și componentele de vizualizare, iar partea server va gestiona logica de business, validarea datelor și interacțiunea cu baza de date. Accesul va fi posibil printr-un browser modern, fără a necesita instalarea de software suplimentar.

Proiectarea se axează pe: scalabilitate, ușurință în utilizare, modularitate, interactivitate și posibilitatea de extindere a funcționalităților în viitor. Sistemul este conceput să sprijine atât analiza individuală, cât și procesul decizional în contexte educaționale, administrative sau de cercetare.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

* Sistemul va fi accesat printr-un browser web modern (ex. Chrome, Firefox, Edge), de pe dispozitive desktop sau laptop.
* Utilizatorii finali vor avea un nivel de bază de alfabetizare digitală, suficient pentru a interacționa cu o interfață grafică web intuitivă.
* Datele socio-demografice utilizate vor fi preluate din surse oficiale (ex. INS, Eurostat), în formate compatibile (CSV, JSON, API).
* Sistemul va rula pe un mediu server Linux, folosind un stack web (ex. Node.js, Python sau PHP, în funcție de implementare).
* Funcționalitățile pot fi extinse pe viitor pentru a integra și alți indicatori statistici (ex. economici, educaționali).
* Aplicația va fi destinată utilizării publice, dar pot exista și niveluri diferite de acces pentru anumiți utilizatori (ex. administratori vs. public general).
* Se presupune că modificările frecvente ale datelor nu vor afecta semnificativ structura bazei de date, ci doar conținutul.

#### Constrângeri

Proiectarea sistemului este condiționată de o serie de constrângeri tehnice și funcționale, care influențează alegerile de arhitectură, dezvoltare și implementare:

* **Mediu hardware și software**: Sistemul va fi accesat printr-un browser web modern, fără a necesita instalare locală. Backend-ul va fi găzduit pe un server compatibil cu tehnologii precum Node.js, Python sau PHP, în funcție de tehnologia aleasă definitiv. Trebuie asigurată compatibilitatea cu diferite sisteme de operare (Windows, Linux, macOS) din perspectiva utilizatorului.
* **Mediu utilizator final**: Platforma este destinată utilizatorilor non-tehnici, motiv pentru care interfața trebuie să fie intuitivă, ușor de folosit și să permită accesul facil la datele vizualizate, chiar și în lipsa unor cunoștințe avansate de analiză.
* **Resurse**: Proiectul se desfășoară în cadrul unei lucrări de licență, ceea ce limitează accesul la infrastructuri performante și echipe de dezvoltare extinse. Soluția tehnică trebuie să fie fezabilă într-un mediu cu resurse reduse (timp, buget, personal).
* **Cerințe de interoperabilitate și standarde**: Se va urmări respectarea bunelor practici în utilizarea formatelor deschise (CSV, JSON, GeoJSON) și a bibliotecilor standard pentru vizualizare și manipulare a datelor (ex: Leaflet, D3.js, Chart.js).
* **Cerințe de securitate și protecție a datelor**: Chiar dacă sistemul nu colectează date personale, trebuie asigurată integritatea și acuratețea datelor provenite din surse oficiale. Se vor evita scripturi sau extensii care pot compromite funcționarea aplicației.
* **Capacitate de stocare și performanță**: Sistemul trebuie să funcționeze fluent cu un volum moderat de date (zeci sau sute de mii de înregistrări), fără a necesita optimizări complexe sau infrastructuri de tip cloud.
* **Testare și validare**: Testarea va fi realizată manual, pe mai multe browsere, în scenarii simple de utilizare, având în vedere constrângerile de timp și resurse ale proiectului.
* **Licențiere**: Se vor utiliza doar biblioteci și tehnologii open-source, pentru a evita costuri suplimentare sau limitări de distribuție.

#### Riscuri

**Risc 1: Modificări în structura datelor de la sursele oficiale (INS, Eurostat)**  
 *Strategie de reducere:* Arhitectură modulară și documentație completă a codului, care permit adaptarea rapidă la modificări.

**Risc 2: Incompatibilități generate de actualizări majore ale framework-urilor utilizate (React, Django)**  
 *Strategie de reducere:* Utilizarea versiunilor LTS, testare continuă și izolarea componentelor critice.

**Risc 3: Întârzieri în sincronizarea datelor sau indisponibilitatea API-urilor externe**  
 *Strategie de reducere:* Implementarea unui sistem de cache local și mesaje de fallback pentru utilizatori.

**Risc 4: Limitări de performanță pe dispozitive vechi sau conexiuni instabile**  
 *Strategie de reducere:* Optimizarea codului front-end și furnizarea de versiuni simplificate ale vizualizărilor.

**Risc 5: Riscuri de securitate în transmiterea și accesarea datelor**  
 *Strategie de reducere:* Criptare TLS 1.3, autentificare OAuth 2.0 și audit automat al accesului.

## Considerații de proiectare

**1. Standardizarea surselor de date**  
 Sistemul trebuie să gestioneze formate variate de la INS și Eurostat, necesitând un proces unitar de prelucrare și validare a datelor.

**2. Suport pentru dispozitive multiple și rezoluții diferite**  
 Aplicația trebuie să fie complet funcțională pe desktop, tabletă și mobil, adaptându-se automat la diferite dimensiuni de ecran.

**3. Accesibilitate și ușurință în utilizare**  
 Interfața trebuie proiectată intuitiv, respectând standardele WCAG pentru accesibilitate, fără a compromite complexitatea analitică.

**4. Performanță sub sarcină mare**  
 Sistemul trebuie să mențină timpi de răspuns optimi în scenarii de încărcare ridicată sau în prezența unor seturi mari de date.

**5. Securitate fără procesare de date personale**  
 Chiar dacă nu sunt colectate date sensibile, sunt necesare măsuri de criptare și control al accesului pentru a proteja conținutul și infrastructura.

### Obiective și linii directoare (ghiduri)

**Obiective principale:**

* **Accesibilitate pentru utilizatori non-tehnici** – Interfața este concepută pentru a fi intuitivă și ușor de învățat, chiar și fără pregătire tehnică.
* **Vizualizare interactivă și clară a datelor** – Accent pe reprezentări grafice (hărți, grafice) care facilitează înțelegerea rapidă a informației.
* **Modularitate și întreținere ușoară** – Arhitectură orientată pe componente, ușor de extins sau modificat.

**Linii directoare tehnice:**

* **Performanță peste consum de resurse** – Se prioritizează viteză de răspuns și timp redus de încărcare, chiar dacă presupune optimizări suplimentare ale memoriei.
* **Ghiduri de codare** – Se respectă convențiile PEP8 (pentru backend Python) și best practices React (pentru frontend), pentru claritate și colaborare eficientă în echipă.
* **UI/UX unificat și responsiv** – Designul urmează un ghid de stil documentat, asigurând consistență vizuală și funcțională pe toate platformele.

**Politici de design:**

* **Preferință pentru tehnologii open-source** – Alegerea unor tehnologii gratuite, mature și bine susținute (React, Django, PostgreSQL).
* **Compatibilitate cu formate comune** – Exportul și importul datelor se face în formate standard (CSV, PDF, JSON) pentru interoperabilitate.
* **Mesaje explicite și feedback vizual** – Sistemul oferă mesaje clare în caz de eroare sau succes, evitând jargonul tehnic.

### Metode de dezvoltare

Sistemul a fost dezvoltat folosind o abordare **orientată pe obiecte**, combinată cu **prototipare incrementală**, pentru a valida din timp interfața și funcționalitățile cheie.

Modelarea arhitecturii și cerințelor s-a realizat cu ajutorul **UML**, utilizând diagrame de caz de utilizare, de clase și de secvență.

Dezvoltarea propriu-zisă a urmat principiile **agile**, cu iterări scurte și testare continuă, adaptând rapid componentele în funcție de feedback și evoluția cerințelor.

**Tehnologii utilizate:**

* Backend: Python + Django (modelare și procesare orientată pe obiecte)
* Frontend: React + Redux (component-based, modular)
* Bază de date: PostgreSQL, modelare relațională

Au fost luate în considerare metode formale pentru validarea cerințelor (ex. verificare prin prototipuri interactive) și documentare automată a interfețelor API.

**Contingente posibile:**

* **Instabilitatea surselor de date externe** (modificări API) – gestionată prin interfețe abstracte și cache local.
* **Lipsa acordurilor privind sincronizarea cu date oficiale** – soluție de rezervă: import manual din fișiere CSV.
* **Posibile incompatibilități între module** – prevenite prin testare integrată și separare clară a responsabilităților între componente.

### Strategii de arhitectură

Sistemul adoptă o arhitectură pe trei straturi: prezentare (React), logică (Django) și stocare (PostgreSQL), pentru claritate și scalabilitate. Comunicarea se face prin API-uri REST, asigurând integrare ușoară cu alte aplicații.

S-a optat pentru tehnologii open-source, stabile și documentate, evitând soluțiile comerciale sau monolitice. Gestionarea erorilor include mesaje explicative și fallback la date cache-uite.

Persistența datelor este locală, cu actualizări programate din surse externe. Arhitectura este gândită modular, permițând adăugarea facilă de noi funcționalități sau surse de date.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Sistemul este împărțit în trei componente principale: interfața utilizator (frontend React), serverul de aplicație (backend Django) și baza de date (PostgreSQL). Fiecare componentă are responsabilități clare și colaborează prin API-uri REST.

Frontend-ul gestionează interacțiunile utilizatorului și vizualizarea datelor, backend-ul procesează cererile, iar baza de date stochează indicatorii socio-demografici. Datele sunt preluate periodic din surse externe (INS, Eurostat), apoi procesate și puse la dispoziția utilizatorului final.

Această decompunere modulară permite o dezvoltare separată a fiecărui strat, testare eficientă și întreținere facilă. S-a evitat o arhitectură monolitică pentru a permite scalabilitate și integrare ulterioară cu alte platforme.

Se utilizează tipare de proiectare precum **MVC** (Model-View-Controller) și **Repository Pattern**, pentru claritate în organizarea codului și separarea logicii de date.

### Vedere logică

Vederea logică evidențiază structura conceptuală a aplicației, organizată în trei straturi:

* **Prezentare:** componente UI React, responsabile de interacțiunea cu utilizatorul.
* **Logică de aplicație:** servicii Django care procesează cererile, gestionează regulile de business și comunică cu baza de date.
* **Date:** PostgreSQL, utilizată pentru stocarea structurată a indicatorilor și metadatelor asociate.

Componentizarea logică permite dezvoltare paralelă, testare modulară și extindere facilă.

### Arhitectură hardware

Sistemul utilizează o arhitectură **centralizată**, găzduită pe un server virtual (VPS) în cloud. Componentele hardware implicate sunt următoarele:

* **Server aplicație și bază de date** – Instanță unică, cu rol dual (aplicație + stocare), configurată cu:
  + CPU: 4 core @ 2.5 GHz
  + RAM: 8 GB
  + SSD: 100 GB
* **Firewall software** – configurat pentru filtrarea traficului pe porturile HTTP (80) și HTTPS (443).
* **Acces securizat** – prin SSH pe portul 22, restricționat la adrese IP cunoscute.

Nu sunt necesare echipamente fizice dedicate, iar sistemul nu implică load balancere sau dispozitive periferice.

### Arhitectură software

**Frontend (strat prezentare):**

* **React 18** – bibliotecă JavaScript pentru UI interactiv și componentizare.
* **Redux** – gestionarea stării aplicației în mod centralizat.
* **Chart.js / D3.js** – biblioteci pentru vizualizări grafice.

**Backend (strat logic):**

* **Python 3.11 + Django** – framework MVC pentru dezvoltare rapidă, clară și sigură.
* **Django REST Framework** – expunerea de API-uri REST JSON pentru interacțiune cu frontend-ul.

**Bază de date (strat date):**

* **PostgreSQL 15** – sistem de gestiune relațională, optimizat pentru interogări complexe și suport GIS.
* **PgAdmin 4** – interfață web pentru administrare și monitorizare.

**Alte componente software:**

* **Docker** – containerizare pentru portabilitate și izolare.
* **Gunicorn + Nginx** – servere pentru livrarea aplicației web.
* **Sistem de operare:** Ubuntu Server 22.04 LTS.

**API-uri externe:**

* **INS & Eurostat** – furnizează date socio-demografice (prin JSON/CSV).
* **Google Maps API** – pentru hărți interactive și geolocalizare.
* **OAuth 2.0 / JWT** – pentru autentificare securizată a utilizatorilor.

Arhitectura software respectă principiile SOLID și utilizează tipare MVC și Singleton, acolo unde este necesar. Diagrama de arhitectură software este inclusă în Anexa E.

### Arhitectura informațiilor

Sistemul stochează date socio-demografice oficiale, precum: populație, natalitate, educație, ocupare, distribuție pe județe și ani. Nu sunt gestionate date cu caracter personal sau informații sensibile.

**Structura datelor:**

* Format: electronic, importate automat din fișiere CSV și API-uri JSON.
* Furnizori: Institutul Național de Statistică (INS), Eurostat.
* Tipuri de date: valori agregate la nivel de regiune, județ, categorie socio-economică și an calendaristic.

**Utilizare internă:**

Datele sunt procesate pentru vizualizare, comparații și generare de rapoarte. Structura bazei de date permite interogări eficiente și export personalizat.

### Arhitectura de comunicații interne

Sistemul funcționează într-un mediu cloud, pe o arhitectură centralizată. Componentele comunicațional conectate sunt:

* **Frontend (React)** comunică cu backend-ul prin HTTPS (port 443), folosind API-uri REST.
* **Backend (Django)** accesează baza de date PostgreSQL prin conexiune locală securizată (port 5432).
* **Firewall software** controlează accesul extern și intern între componente, permițând doar porturile necesare (443, 5432, 22).

**Echipamente fizice suplimentare nu sunt necesare.**  
 Arhitectura de comunicații este complet virtualizată în cadrul unui VPS, cu suport pentru rețea privată virtuală (VPN) dacă este necesară extinderea.

**Estimări de rețea:**

* Trafic LAN intern estimat: ~10 MB/s pentru încărcări de date și generare rapoarte.
* Trafic WAN estimat: până la 100 MB/zi, în funcție de numărul de utilizatori și actualizări de date externe.

### Diagrama de arhitectură a sistemului

## Proiectarea sistemului

### Proiectarea bazei de date

Baza de date utilizată este **PostgreSQL**, structurată relațional pentru a permite interogări complexe și performanță optimă. Datele sunt organizate în tabele tematice, corelate prin chei primare și străine.

**Tabele principale:**

* indicatori – conține denumiri și descrieri ale indicatorilor socio-demografici.
* valori\_indicatori – stochează valorile numerice ale indicatorilor, împreună cu anul, regiunea și unitatea de măsură.
* regiuni – include informații despre județe și codurile administrative.
* utilizatori – pentru conturi înregistrate (opțional, dacă se folosesc funcționalități de salvare/export).

**Format date:** electronic (CSV/JSON importat), validat automat înainte de inserare.  
 **CRUD:** complet – sistemul permite creare, citire, actualizare și ștergere pentru datele interne.  
 **Validare:** se verifică unicitatea perechilor indicator-regiune-an și formatul valorilor numerice.  
 **Stocare:** SSD performant, cu backup zilnic automat.

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

Sistemul utilizează următoarele obiecte de date funcționale esențiale:

* **Indicator socio-demografic**  
   Structură: { id, denumire, descriere, categorie }  
   Rol: identifică tipul de date stocate (ex: natalitate, populație).
* **Valoare indicator**  
   Structură: { id, id\_indicator, id\_regiune, an, valoare, unitate\_masura }  
   Rol: păstrează valorile statistice efective înregistrate.
* **Regiune**  
   Structură: { id, nume, cod\_iso, tip (județ/municipiu), coordonate }  
   Rol: definește zona geografică pentru care sunt raportate datele.
* **Utilizator (opțional)**  
   Structură: { id, email, parola\_hash, rol, preferinte\_export }  
   Rol: permite personalizarea experienței de utilizare.

**Transfer de date între componente:**  
 Frontend-ul primește structuri JSON conținând date agregate (indicator + valori + metadate), utilizate în hărți, grafice și tabele. Structurile sunt optimizate pentru încărcare rapidă și randare vizuală.

#### Fișiere și baze de date

**Model de date fizic:**  
 Datele sunt stocate într-o bază de date relațională PostgreSQL, organizată în tabele normalizate. Structura este optimizată pentru interogări rapide și relații clare între indicatori, regiuni și ani.

**Structuri principale:**

* Tabele SQL: indicatori, valori\_indicatori, regiuni, utilizatori
* Indexuri: pe coloanele an, id\_indicator, id\_regiune pentru performanță

**Fișiere de sistem asociate:**

* **Import:** fișiere CSV pentru încărcarea datelor brute (INS, Eurostat)
* **Export:** fișiere PDF/Excel generate temporar pentru rapoarte
* **Loguri:** fișiere text pentru monitorizarea operațiunilor, salvate local

**Locație fișiere:**

* /data/import/ – fișiere brute CSV
* /data/export/ – rapoarte generate la cerere
* /logs/ – fișiere de activitate și erori

Datele sunt salvate pe SSD, cu backup automat zilnic în cloud. Accesul este controlat prin permisiuni la nivel de server.

### Conversii de date

Sistemul include un modul automat de conversie a datelor brute primite de la surse externe (INS, Eurostat) în format compatibil cu structura internă a bazei de date.

**Conversii efectuate:**

* Din **CSV** → tabele SQL (cu validare și mapare a coloanelor)
* Din **API JSON** → obiecte Python, apoi inserare în baza de date
* Conversii de tipuri: string → integer/float/date, acolo unde este necesar
* Uniformizarea unităților de măsură (ex: % în valori absolute, unde se aplică)

**Documentația completă** privind conversiile este stocată în directorul tehnic al proiectului, la:  
 /docs/conversii\_date.md

### Interfețe utilizator

Sistemul definește următoarele clase de utilizatori:

* **Vizitator anonim**  
   *Acces:* doar vizualizare hărți și grafice  
   *Competență:* minimă  
   *Estimare utilizatori:* majoritatea traficului (~70%)  
   *Simultan:* până la 50 utilizatori
* **Utilizator înregistrat**  
   *Acces:* vizualizare + salvare rapoarte personalizate  
   *Competență:* medie (studenți, cercetători)  
   *Estimare:* 20–30% din utilizatori  
   *Simultan:* până la 20 utilizatori
* **Administrator**  
   *Acces:* gestionare date, utilizatori, rapoarte  
   *Competență:* avansată  
   *Estimare:* 1–2 persoane  
   *Simultan:* 1 utilizator maxim

Sistemul este proiectat pentru maximum **100 de utilizatori simultan**, dintre care ~70% pot fi externi (public general).

#### Intrări

Utilizatorii interacționează cu sistemul exclusiv prin interfața web, folosind următoarele mijloace de introducere a datelor:

* **Selectoare de indicatori, ani și regiuni** – meniuri derulante (dropdown)
* **Câmpuri de căutare** – pentru filtrare rapidă după denumire sau cod
* **Butoane de acțiune** – pentru generare hartă, export, comparare

**Fluxul de date:**

* Se selectează un indicator → se declanșează încărcarea datelor aferente din baza de date
* Schimbarea anului sau regiunii actualizează vizualizarea în timp real

**Validări la introducere:**

* Indicatori și ani – doar din listă predefinită
* Obligatorii: toți parametrii vizuali (indicator, an, regiune)
* Interval numeric pentru datele manuale (doar administratori): valori între 0 și 1000000, zecimale permise (max 2)

**Mesaje și controale:**

* Feedback vizual la selecție (culori, animații)
* Mesaje de eroare: „Date indisponibile”, „Selecție invalidă”
* Prevenire dublă introducere: blocare butoane până la finalizarea cererii

**Aspect GUI:**  
 Ecranele sunt responsive, structurate în: panou de selecție (stânga), hartă/grafice (centru), panou cu rezultate/export (dreapta). Toate componentele sunt documentate în ghidul UI (Anexa UI/UX).

#### Ieșiri

Sistemul generează următoarele tipuri de ieșiri:

* **Ecrane de afișare (vizualizări web):**
  + **Cod:** UI-MAP-01  
     *Conținut:* hartă interactivă cu indicatori colorați pe județe  
     *Elemente:* indicator, an, valoare județeană, legendă, tooltip
  + **Cod:** UI-CHART-01  
     *Conținut:* grafice comparative (bară, linie, pie)  
     *Elemente:* etichete axă, valori, comparații procentuale
* **Rapoarte exportabile:**
  + **Cod:** RPT-DEMOG-01  
     *Format:* PDF / Excel  
     *Conținut:* tabele statistice, grafice incluse, sumar observații  
     *Utilizatori:* cercetători, studenți, ONG-uri  
     *Acces:* doar utilizatori înregistrați
* **Mesaje sistem:**
  + Pop-up-uri informative (ex: „Export realizat cu succes”)
  + Mesaje de eroare (ex: „Indicatorul nu are date pentru anul selectat”)

**Scopul ieșirilor:** prezentarea clară a informațiilor statistice și susținerea deciziilor sau cercetărilor prin date vizualizate și documentate.

### Proiectarea interfețelor cu utilizatorul

Interfața aplicației este web-based, proiectată pentru claritate, accesibilitate și interactivitate. Designul este responsive și se adaptează automat la dispozitive desktop, tabletă și mobil.

**Componente principale:**

* **Panou de selecție (stânga):**
  + Dropdown-uri pentru indicatori, ani, regiuni
  + Căutare rapidă cu sugestii inteligente
* **Zonă centrală de vizualizare:**
  + Hartă tematică interactivă
  + Grafice dinamice și comparări multiple
  + Feedback vizual instant (hover, tooltip, animații)
* **Panou de rezultate (dreapta):**
  + Statistici detaliate
  + Butoane pentru export și salvare rapoarte
  + Informații suplimentare/contextuale

**Principii de design aplicate:**

* Contrast ridicat și fonturi min. 14px pentru lizibilitate
* Elemente interactive cu efecte de hover și validare vizuală
* Navigație intuitivă și minimă încărcare cognitivă

**Accesibilitate:** conform WCAG 2.1 nivel AA – testat pentru cititoare de ecran și compatibil cu navigare prin tastatură.

Ghidul complet de stil UI/UX este inclus în Anexa UI.

## Scenarii de utilizare

#### **Scenariul 1 – Vizualizare interactivă (Utilizator anonim)**

**Context:** Un vizitator accesează platforma pentru a analiza rata natalității în România.  
 **Pași:**

1. Accesează aplicația prin browser.
2. Selectează indicatorul "Rata natalității".
3. Alege anul 2021.
4. Harta tematică este generată automat.
5. Plasează cursorul peste un județ → apare tooltip cu valoarea exactă.
6. Schimbă anul din 2021 în 2015 → harta se actualizează în timp real.

**Rezultat:** Utilizatorul observă evoluția indicatorului pe regiuni și în timp, fără a fi necesară autentificarea.

#### **Scenariul 2 – Export raport personalizat (Utilizator înregistrat)**

**Context:** Un cercetător dorește să compare nivelul educației între două județe.  
 **Pași:**

1. Se autentifică în platformă.
2. Selectează indicatorul "Nivel educațional mediu".
3. Alege județele Cluj și Dolj, pentru anii 2019–2023.
4. Aplicația generează un grafic comparativ.
5. Apasă „Exportă raport” → se generează un fișier PDF.
6. Fișierul este descărcat local.

**Rezultat:** Utilizatorul obține un raport detaliat, gata de inclus în documentație academică.

#### **Scenariul 3 – Actualizare date (Administrator)**

**Context:** Administratorul actualizează baza de date cu noi statistici.  
 **Pași:**

1. Se conectează în interfața de administrare.
2. Încarcă un fișier CSV cu date noi pentru anul 2024.
3. Sistemul validează structura fișierului.
4. Dacă nu sunt erori, datele sunt inserate automat.
5. Utilizatorii pot vizualiza noile date în aplicație.

**Rezultat:** Sistemul este actualizat cu cele mai recente informații, fără întreruperi.

## Proiectare de detaliu

Proiectarea de detaliu oferă specificațiile necesare pentru implementarea sistemului, integrarea componentelor și livrarea unui produs funcțional.

**Componente software:**

* **Frontend (React + Redux):**
  + Structurat pe componente reutilizabile.
  + Integrare cu API-ul prin fetch/axios.
  + Responsivitate asigurată prin media queries și framework CSS modular.
* **Backend (Django + DRF):**
  + Organizare pe aplicații: vizualizare, comparatie, autentificare.
  + Endpoint-uri REST standardizate (ex: /api/indicatori/, /api/valori/).
  + Validări de date incluse în serializer-ele DRF.
* **Baza de date (PostgreSQL):**
  + Scripturi SQL inițiale pentru creare tabele și populare cu date.
  + Indexuri și constrângeri definite pentru optimizare.

**Integrare hardware-software:**

* Aplicația rulează pe un VPS configurat cu Ubuntu 22.04.
* Componentele sunt containerizate cu Docker și orchestrate prin Docker Compose.
* Serverul Nginx asigură rutarea traficului și securizarea cu HTTPS.

**Flux de dezvoltare:**

* Controlul versiunilor prin Git.
* Testare unitară și de integrare (Pytest, Jest).
* Automatizare CI/CD cu GitHub Actions.

Această structură permite dezvoltatorilor să implementeze rapid funcționalitățile, să integreze module și să testeze complet sistemul înainte de lansare.

### Proiectare hardware de detaliu

Sistemul rulează pe o infrastructură cloud (VPS) care nu necesită componente hardware fizice dedicate la nivelul utilizatorului final. Totuși, sunt definite următoarele specificații minime pentru componenta de server:

#### **1. Server virtual (VPS) – producător recomandat: Hetzner / DigitalOcean**

* **Procesor:** 4 vCPU @ 2.5 GHz (minim)
* **RAM:** 8 GB DDR4
* **Stocare:** SSD 100 GB
* **Rețea:** conexiune garantată 100 Mbps
* **Sistem operare:** Ubuntu Server 22.04 LTS
* **Alimentare:** alimentare redundantă (asigurată de furnizor)

#### **2. Dispozitive externe opționale (pentru testare locală)**

* **Laptop/PC utilizator:**
  + CPU: min. dual-core @ 2.0 GHz
  + RAM: min. 4 GB
  + Rezoluție ecran: min. 1366x768
  + Browser compatibil: Chrome, Firefox, Edge (ultimele 2 versiuni)
* **Imprimantă (pentru rapoarte):**
  + Suport A4
  + Conectare: USB sau Wi-Fi

#### **3. Specificații de conectivitate**

* Toate interacțiunile se fac prin HTTPS (port 443)
* Acces administrativ prin SSH (port 22)
* Conexiuni interne între aplicație și baza de date pe portul 5432 (PostgreSQL)

### Proiectare software de detaliu

#### **1. Identificator serviciu: api/indicatori/**

**Clasificare:** Serviciu REST – date  
 **Definiție:** Returnează lista completă de indicatori socio-demografici disponibili în sistem  
 **Cerințe:** CF1 – selectare indicator; CNF – timp răspuns < 2 sec.  
 **Structuri de date interne:** Model Django Indicator → {id, denumire, descriere, categorie}  
 **Constrângeri:** doar citire (GET); necesită validare JSON  
 **Compoziție:** utilizează serializer IndicatorSerializer  
 **Utilizatori/Interacțiuni:** accesat de frontend la inițializare pagină sau selectare meniu  
 **Procesare:** extragere din baza de date, sortare alfabetică, serializare JSON  
 **Interfețe/Exporturi:** Endpoint public; nu necesită autentificare  
 **Raportare design:** ~1–2 KB per răspuns

#### **2. Identificator serviciu: api/valori/**

**Clasificare:** Serviciu REST – date  
 **Definiție:** Returnează valorile pentru un indicator dat, pe regiuni și ani  
 **Cerințe:** CF2, CF3 – afișare și comparație valori; CNF – max. 3 sec. la încărcare  
 **Structuri de date interne:** Model ValoareIndicator → {id\_indicator, id\_regiune, an, valoare}  
 **Constrângeri:** validare existență indicator și an; fallback la cel mai apropiat an dacă lipsesc datele  
 **Compoziție:** include serializer și agregator statistic pentru comparare  
 **Utilizatori/Interacțiuni:** utilizat la generarea hărților și grafice comparate  
 **Procesare:** interogare filtrată, grupare pe regiuni, calcul procentual (dacă este cazul)  
 **Interfețe/Exporturi:** REST JSON, accesibil doar prin cerere POST validă  
 **Raportare design:** răspuns mediu: 5–10 KB

#### **3. Identificator serviciu: api/export/pdf/**

**Clasificare:** Serviciu aplicație – export  
 **Definiție:** Generează raport PDF cu grafice și statistici selectate  
 **Cerințe:** CF4 – export personalizat; CNF – timp generare < 10 sec.  
 **Structuri de date interne:** structuri temporare pentru layout PDF, incluzând grafic generat ca imagine  
 **Constrângeri:** doar pentru utilizatori autentificați; max 5 MB per raport  
 **Compoziție:** integrează librăria ReportLab (Python)  
 **Utilizatori/Interacțiuni:** utilizatori înregistrați → dashboard personalizat  
 **Procesare:** primire parametri → generare layout → conversie grafic în imagine → compilare PDF  
 **Interfețe/Exporturi:** livrare ca fișier descărcabil  
 **Raportare design:** 500 KB – 2 MB / raport

#### **4. Identificator serviciu: auth/login/ și auth/register/**

**Clasificare:** Servicii de autentificare  
 **Definiție:** Gestionează login și înregistrare utilizatori  
 **Cerințe:** CNF – securitate conform OAuth 2.0; criptare parole  
 **Structuri de date interne:** model Utilizator + JWT pentru token  
 **Constrângeri:** parole min. 8 caractere; token valabil 24h  
 **Compoziție:** serializer, validator, manager de sesiuni  
 **Utilizatori/Interacțiuni:** utilizatori înregistrați + admini  
 **Procesare:** validare credențiale → generare token → stocare în cookie HTTP-only  
 **Interfețe/Exporturi:** API REST + middleware de protecție rute  
 **Raportare design:** răspuns <1 KB; JWT ~300 B

### Proiectare detaliată de securitate

Sistemul implementează măsuri de securitate standard pentru a proteja datele și funcționalitățile aplicației.

#### **Autentificare**

* Se utilizează mecanismul **OAuth 2.0** cu token-uri **JWT (JSON Web Tokens)**.
* Autentificarea se realizează prin API-ul auth/login/ folosind email și parolă.
* Token-ul este stocat în cookie HTTP-only și are valabilitate limitată (24h).

#### **Autorizare**

* Există trei niveluri de acces:
  + **Vizitator anonim** – acces doar la vizualizare.
  + **Utilizator autentificat** – acces la funcții suplimentare (export, salvare).
  + **Administrator** – acces complet la gestionarea datelor și utilizatorilor.
* Rutele API sunt protejate prin decorators de permisiune în backend (Django REST Framework).

#### **Jurnalizare și auditare**

* Toate acțiunile critice (login, export, modificări date) sunt înregistrate în loguri text.
* Logurile sunt arhivate zilnic și păstrate timp de 6 luni.
* Accesul la loguri este restricționat la administratori.

#### **Criptare**

* **TLS 1.3** este activ pe toate canalele de comunicație externe (HTTPS).
* Parolele sunt criptate cu algoritmul **PBKDF2** + salt.
* Token-urile JWT sunt semnate cu cheie secretă internă (HMAC SHA-256).

#### **Utilizarea porturilor de rețea**

* Porturi utilizate:
  + **443** – HTTPS (frontend & API)
  + **22** – SSH (acces administrativ, restricționat)
  + **5432** – PostgreSQL (intern, accesat doar de backend)
* Restul porturilor sunt închise prin firewall.

#### **Detectare și prevenire a intruziunilor**

* Serverul folosește fail2ban pentru blocarea adreselor IP suspecte.
* Se generează alerte automate pentru 5+ încercări eșuate de autentificare.
* Monitorizarea activă se face prin instrumente externe (ex. UptimeRobot, Netdata).

### Proiectare de detaliu pentru performanța sistemului

#### **Cerințe/Estimări de capacitate și volum**

* **Utilizatori activi simultan:** max. 100
* **Volum date gestionate:** ~500.000 înregistrări în baza de date
* **Dimensiune totală a bazei de date:** ~200 MB (în creștere anuală)

#### **Așteptări de performanță**

* **Timp de răspuns interogări simple (ex: 1 județ):** < 2 sec
* **Timp de răspuns interogări complexe (ex: comparații multi-anuale):** < 5 sec
* **Timp generare raport PDF:** < 10 sec
* **Încărcare hartă interactivă:** < 3 sec

#### **Cerințe de disponibilitate**

* **Disponibilitate țintă:** 99.5%
* **Fereastră de mentenanță:** noaptea, cu notificare prealabilă
* **Timp maxim de nefuncționare admis lunar:** ~3 ore

#### **Proiectare de performanță**

* Indexuri aplicate pe coloanele an, indicator, regiune
* Cache local pentru interogările frecvente
* Frontend optimizat cu lazy-loading și preîncărcare asincronă
* Server configurat cu alocare dedicată de resurse (CPU + RAM garantate)

#### **Proiectare de fiabilitate**

* Serviciile rulează în containere izolate (Docker) pentru izolarea erorilor
* Aplicația web repornește automat în caz de cădere (watchdog activ)
* Sistemul este monitorizat permanent (Netdata / UptimeRobot)

#### **Backup, recuperare și arhivare**

* Backup zilnic automat al bazei de date (stocare criptată, off-site)
* Loguri și rapoarte arhivate lunar
* Procedură de restaurare completă testată periodic

#### **Puncte unice de eșec și măsuri**

* **Punct critic:** serverul VPS unic
* **Măsură de rezervă:** backup complet disponibil pentru restaurare în <1h
* **Opțional:** posibil upgrade la arhitectură cu load balancing și replicare BD pentru disponibilitate ridicată (HA)

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

Sistemul este compus din trei componente principale: frontend (React), backend (Django REST API) și baza de date (PostgreSQL), găzduite într-o arhitectură **centralizată**, pe un **singur server VPS**.

#### **Număr de servere și clienți**

* **1 server aplicație (backend + frontend)** – găzduit într-un container Docker
* **1 server de baze de date (PostgreSQL)** – container separat, pe același VPS
* **Clienți:** nelimitați, prin browser (web-based)

#### **Cerințe de sincronizare și control**

* Nu se aplică sincronizare de tip hardware (bus) – componentele comunică asincron prin protocoale HTTP/HTTPS în rețea locală virtuală (Docker bridge)
* Comunicarea este **stateless**, cu cereri HTTP REST și răspunsuri JSON

#### **Formate de date schimbate între componente**

* **Frontend → Backend:** JSON (ex: parametri selectați: indicator, an, regiune)
* **Backend → Frontend:** JSON (ex: liste de valori, structuri pentru hărți/grafice)
* **Backend → Bază de date:** SQL / ORM Django
* **Bază de date → Backend:** rezultate SQL serializate în JSON

#### **Conectivitate și direcția fluxului de date**

* **Frontend** → trimite cereri către **Backend** (HTTP POST/GET)
* **Backend** → interoghează **PostgreSQL** (conexiune locală pe port 5432)
* **Frontend ← Backend:** primește date prelucrate pentru afișare

*Toate comunicațiile între containere sunt izolate în rețea Docker, fără expunere externă între backend și bază de date.*

#### **Topologie LAN (internă, virtuală)**

* Topologie: **punct-la-punct** (backend ↔ DB), **client-server** (frontend ↔ backend)
* Rețea Docker locală (bridge) cu DNS automatizat pentru containere
* Distanță fizică: irelevantă (toate componentele rulează pe același host)

## Controale pentru verificarea integrității sistemului

#### **1. Securitate internă**

* Accesul la datele critice este controlat prin roluri definite în sistem: **utilizator**, **administrator**.
* Operațiile CRUD (creare, modificare, ștergere) asupra bazei de date sunt permise doar administratorilor autentificați.
* API-urile sensibile sunt protejate prin permisiuni (decorators) în backend-ul Django REST Framework.

#### **2. Proceduri de audit**

* Toate acțiunile majore (login, adăugare date, export) sunt logate în fișiere securizate.
* Logurile sunt arhivate zilnic și păstrate timp de **6 luni**, conform politicilor interne.
* Rapoartele de export sunt marcate cu **ID de utilizator, dată și oră**.

#### **3. Piste de auditare a aplicației**

* Pentru fiecare acțiune critică se salvează:
  + ID utilizator
  + Adresa IP (terminal rețea)
  + Tip acțiune (ex: modificare date, accesare fișier export)
  + Data și ora exactă
* Sistemul oferă funcționalitate de **reconstituire a activității utilizatorilor** în scop de audit.

#### **4. Validarea datelor**

* Câmpurile introduse de utilizator sunt validate automat:
  + Valori numerice între limite predefinite
  + Formate standardizate pentru ani, denumiri, unități de măsură
  + Tabele de validare interne pentru regiuni și indicatori (referințe externe standardizate)

#### **5. Procese de verificare a operațiilor critice**

* Fiecare operație de adăugare/modificare/ștergere:
  + Este validată la nivel de backend
  + Este confirmată de sistem înainte de executare (ex: „Sigur doriți să ștergeți?”)
  + Este jurnalizată complet pentru urmărire ulterioară

#### **6. Identificare și trasabilitate completă**

* Fiecare intrare în log conține:
  + **ID utilizator**
  + **Adresa IP (sau identificator terminal)**
  + **Timestamp complet (data + oră exactă)**
  + **Operația efectuată**
  + **Resursa accesată sau modificată**

Anexa A: Gestiunea modificărilor documentului

|  |  |  |  |
| --- | --- | --- | --- |
| **Versiune** | **Data** | **Autorul/Deținătorul** | **Descriere modificare** |
| 1.0 | 10/04/2025 | Aurică Ciprian | Versiune inițială a documentului |
| 1.1 | 11/04/2025 | Aurică Ciprian | Adăugare secțiuni: 3.2, 4.2, 5.3.2 |
| 1.2 | 13/04/2025 | Aurică Ciprian | Adăugare și structurare Anexe A–C |

Anexa B: Acronime

|  |  |
| --- | --- |
| **Acronim** | **Forma completă** |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| CSV | Comma-Separated Values |
| DRF | Django REST Framework |
| EU | European Union |
| INS | Institutul Național de Statistică |
| IP | Internet Protocol |
| JWT | JSON Web Token |
| PDF | Portable Document Format |
| REST | Representational State Transfer |
| UI | User Interface |
| UX | User Experience |
| VPS | Virtual Private Server |
| WCAG | Web Content Accessibility Guidelines |

Anexa C Documente la care se face referire

|  |  |  |
| --- | --- | --- |
| **Nume document** | **Locație sau URL** | **Dată emitere document** |
| Documentul de cerințe (SRS) – Laborator 1 | /docs/ Documentul de cerințe (SRS) – Laborator 1 | 27/02/2025 |
| Ghid proiectare aplicații web (intern) | /docs/ghid\_proiectare\_web.pdf | 10/03/2025 |
| Manual Django REST Framework | [https://www.django-rest-framework.org](https://www.django-rest-framework.org/) | Continuu Actualizat. |
| Documentație API Eurostat | <https://ec.europa.eu/eurostat/web/json-and-unicode-web-services> | 01/02/2025 |