**APLICAȚIE PENTRU PREDICȚIA PREȚURILOR DIN DOMENIUL IMOBILIAR**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Larisa-Elena DUD** |
|  |  |  |
|  | Coordonator științific: | **Prof. dr. ing. Gheorghe SEBESTYEN-PAL** |

**2023**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DECAN, |  | | DIRECTOR DEPARTAMENT, | |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Larisa-Elena DUD**

**APLICAȚIE PENTRU PREDICȚIA PREȚURILOR DIN DOMENIUL IMOBILIAR**

1. **Enunțul temei:** *Scurtă descriere a temei lucrării de licență și datele inițiale*
2. **Conținutul lucrării:** *Introducere, Obiectivele proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexe*
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare.
4. **Consultanți**: Prof. dr. Ing. Gheorghe SEBESTYEN-PAL
5. **Data emiterii temei:** 1 noiembrie 2022
6. **Data predării:** 5 septembrie 2023

|  |  |
| --- | --- |
| Absolvent: | Larisa-Elena DUD |
|  |  |
| Coordonator științific: | Prof. fr. Ing. Gheorghe SEBESTYEN |

**Declarație pe propria răspundere privind**

**autenticitatea lucrării de licență**

Subsemnata DUD LARISA-ELENA, legitimată cu CI seria HD nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, autorul lucrării \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universității Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

|  |  |  |
| --- | --- | --- |
| Data  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

Cuprins

Contents

[Capitolul 1. Introducere 1](#_Toc144367908)

[1.1. Contextul proiectului 1](#_Toc144367909)

[1.2. Motivația alegerii lucrării de licență 1](#_Toc144367910)

[1.3. Structura lucrării 2](#_Toc144367911)

[Capitolul 2. Obiectivele proiectului 3](#_Toc144367912)

[2.1. Obiective principale 3](#_Toc144367913)

[2.2. Obiective secundare 4](#_Toc144367914)

[Capitolul 3. Studiu bibliografic 5](#_Toc144367915)

[3.1. Test 5](#_Toc144367916)

[Capitolul 4. Analiză și fundamentare Teoretică 6](#_Toc144367917)

[4.1. Modelul de predicție a prețului 6](#_Toc144367918)

[4.1.1. Seturile de date utilizate 6](#_Toc144367919)

[4.1.2. Preprocesarea datelor 6](#_Toc144367920)

[4.1.3. Algoritmi de codificare a locațiilor 7](#_Toc144367921)

[4.1.4. Utilizarea regresiei liniare pentru predictia pretului 8](#_Toc144367922)

[4.1.5. Utilizarea Random Forest pentru predictia pretului 8](#_Toc144367923)

[4.1.6. Utilizarea clasificatorului Support Vector Machines pentru predictia pretului 9](#_Toc144367924)

[4.1.7. Algoritmul de detectare a anomaliilor 9](#_Toc144367925)

[4.2. Colectarea datelor de pe internet 10](#_Toc144367926)

[4.3. Structura logică a aplicației 10](#_Toc144367927)

[4.4. Medii de programare 10](#_Toc144367928)

[Capitolul 5. Proiectare de detaliu și implementare 12](#_Toc144367929)

[5.1. Arhitectura generală a aplicației 12](#_Toc144367930)

[5.2. Structura programului 12](#_Toc144367931)

[5.2.1. Diagrame de pachete 12](#_Toc144367932)

[5.2.2. Diagrame de clase 14](#_Toc144367933)

[5.3. Modulul de predicție a prețului 15](#_Toc144367934)

[5.3.1. Clasa loadData 15](#_Toc144367935)

[5.3.2. Clasa Preprocesare 15](#_Toc144367936)

[5.3.3. Clasa de codificare a locatiilor 15](#_Toc144367937)

[5.3.4. Clasa pentru antrenarea modelului 15](#_Toc144367938)

[5.3.5. Clasa pentru calcularea perfomantei modelului 15](#_Toc144367939)

[5.3.6. Clasa pentru detectia si eliminarea anomaliilor 15](#_Toc144367940)

[5.3.7. Clasa pentru generarea graficelor 15](#_Toc144367941)

[5.4. Modulul de colectare a datelor de pe internet 15](#_Toc144367942)

[5.5. Aplicația Web 16](#_Toc144367943)

[5.5.1. Comunicarea între Backend si Frontend 16](#_Toc144367944)

[5.5.2. Componenta de meniu 16](#_Toc144367945)

[5.5.3. Componenta de login 17](#_Toc144367946)

[5.5.4. Componenta de Inregistrare cont 18](#_Toc144367947)

[5.5.5. Pagina de vizualizare a anunturilor 18](#_Toc144367948)

[5.5.6. Componenta administratorului 19](#_Toc144367949)

[5.5.7. Comoponenta de statistici 20](#_Toc144367950)

[5.5.8. Componenta pentru estimarea pretului 20](#_Toc144367951)

[5.5.9. Componenta principală 21](#_Toc144367952)

[5.5.10. Alte componente importante 21](#_Toc144367953)

[5.6. Aplicatia de gestiune a utilizatoriilor 22](#_Toc144367954)

[5.6.1. Descrierea claselor si metodelor 22](#_Toc144367955)

[5.6.1.1. Clasa User 22](#_Toc144367956)

[5.6.1.2. Interfata UserRepository 22](#_Toc144367957)

[5.6.1.3. Clasa UserBuilder 22](#_Toc144367958)

[5.6.1.4. Clasa UserDTO 22](#_Toc144367959)

[5.6.1.5. Clasa LoginDTO 22](#_Toc144367960)

[5.6.1.6. Clasa UserController 22](#_Toc144367961)

[5.6.1.7. Clasa UserService 23](#_Toc144367962)

[5.6.2. Persistenta datelor 23](#_Toc144367963)

[Capitolul 6. Testare și validare 25](#_Toc144367964)

[6.1. Testarea si validarea modelelor de predictie 25](#_Toc144367965)

[6.2. Testarea si validarea modulului care colecteaza anunturi 25](#_Toc144367966)

[Capitolul 7. Manual de instalare si utilizare 26](#_Toc144367967)

[7.1. Resurse necesare 26](#_Toc144367968)

[7.2. Manual de utilizare 26](#_Toc144367969)

[Capitolul 8. Concluzii 31](#_Toc144367970)

[8.1. Contribuții personale 31](#_Toc144367971)

[8.2. Analiza critică a rezultatelor obținute 31](#_Toc144367972)

[8.3. Posibile dezvoltări și îmbunătățiri ulterioare 31](#_Toc144367973)

[Capitolul 9. Bibliography 32](#_Toc144367974)

[Anexa 1 34](#_Toc144367975)

[Anexa 2 35](#_Toc144367976)

[Anexa 3 36](#_Toc144367977)

[Capitolul 1. Introducere 1](#_Toc129864352)

1.1. Contextul proiectului....................................................................................... 1

1.2. Motivația alegerii lucrării de licență............................................................... 1

1.3. Structura lucrării............................................................................................. 2

[Capitolul 2. Obiectivele proiectului 3](#_Toc129864353)

2.1. Obiective principale.........................................................................................3

2.2. Obiective secundare.........................................................................................3

[Capitolul 3. Studiu bibliografic 4](#_Toc129864354)

3.1. Metode și modele de predictie automată..........................................................4

3.2. Tehnici de colectare a datelor de pe internet.....................................................4

3.3. Aplicații existente ............................................................................................4

[Capitolul 4. Analiză și fundamentare Teoretică 6](#_Toc129864355)

4.1. Modelul de predicție a prețului.........................................................................6

4.1.1. Seturile de date utilizate............................................................................6

4.1.2. Preprocesare datelor..................................................................................6

4.1.3. Algoritmi de codificare a locațiilor............................................................6

4.1.4. Regresia Liniara ........................................................................................6

4.1.5. Algoritmul de detectare a anomaliilor........................................................6

4.1.6. Metrica folosita pentru vizualizarea perfomantei.........................................6

4.2. Colectarea datelor de pe Internet......................................................................6

4.3. Tehnici de vizualizare

4.3. Structura logică a aplicației .............................................................................6

4.4. Limbaje utilizate...............................................................................................6

[Capitolul 5. Proiectare de detaliu și implementare 8](#_Toc129864358)

5.1. Arhitectura generală a aplicației..........................................................................8

5.2. Diagrame importante...........................................................................................8

5.2.1. Diagrame de pachete.....................................................................................8

5.2.2. Diagrame de clase.........................................................................................8

5.3. Modelul de predicție a pretului............................................................................8

5.3.1. Funcția de preprocesare a datelor..................................................................8

5.3.2. Funcția pentru codificarea locațiilor.............................................................8

5.3.3. Antrenarea modelului...................................................................................8

5.3.4. Determinarea performanței modelului.........................................................8

5.3.5. Funcția de detecție și eliminare a anomaliilor..............................................8

5.3.4. Generarea graficelor.....................................................................................8

5.4. Modulul de colectare a datelor de pe internet......................................................8

5.5. Aplicația Web......................................................................................................8

5.5.1. Comunicarea între Backend și Frontend.......................................................8

5.5.2. Componenta de Login...................................................................................8

5.5.3. Componenta de Inregistrare cont..................................................................8

5.5.4. Componenta de Anunturi..............................................................................8

5.5.5. Componenta administratorului.....................................................................8

5.5.6. Componenta de statistici...............................................................................8

5.5.7. Componenta principală.................................................................................8

5.5.8. Alte componente importante.........................................................................8

5.6. Aplicația de gestiune a utilizatorilor....................................................................8

5.6.1. Descrierea claselor și a metodelor.................................................................8

5.6.2. Persistența datelor.........................................................................................8

[Capitolul 6. Testare și validare 10](#_Toc129864359)

6.1. Testarea și validarea modelelor de predicție......................................................10

6.1.1. Modelul pentru setul de date din București................................................10

6.1.2. Modelul pentru setul de date din Cluj.........................................................10

6.2. Testarea și validare modulului care colectează anunțuri....................................10

[Capitolul 7. Manual de instalare si utilizare 12](#_Toc129864360)

7.1. Cerințe de sistem................................................................................................12

7.1.1. Cerințe hardware.........................................................................................12

7.1.2. Cerințe software..........................................................................................12

7.2. Manual de instalare............................................................................................12

7.2.1. Instalare Python..........................................................................................12

7.2.2. Instalare Node.js.........................................................................................12

7.2.3. Instalare IntelliJ...........................................................................................12

7.2.4. Instalare Visual Studio Code.......................................................................12

7.2.5. Instalarea dependințelor interne..................................................................12

7.3. Manual de utilizare.............................................................................................12

[Capitolul 8. Concluzii 14](#_Toc129864361)

8.1. Contribuții personale..........................................................................................14

8.2. Analiza critică a rezultatelor obținute................................................................14

8.3. Posibilie dezvoltări și îmbunătățiri ulterioare....................................................15

[Bibliografie 16](#_Toc129864362)

[Anexa 1....................................................................................................18](#_Toc129864363)

[Anexa 2................ 19](#_Toc129864364)

[Anexa 3......... 20](#_Toc129864365)

# Introducere

## Contextul proiectului

Piața imobiliară reprezintă un sector de interes major, care joacă un rol esențial în viața de zi cu zi a oamenilor și în economie. Achiziționarea sau vânzarea unei proprietăți reprezintă deseori o decizie importantă și financiar semnificativă pentru indivizi și companii. În acest context, evaluarea cât mai corectă a proprietăților și întelegerea pieței imobiliare sunt esențiale pentru a asigura o tranzacție eficientă și echitabilă.

Domeniul imobiliar se caracterizează prin complexitate și diversitate, deoarece implicațiile financiare și juridice sunt considerabile. Evaluarea proprietăților imobiliare presupune un proces care implică analiza unei game largi de factori, cum ar fi: localizarea, suprafața, calitatea construcției, facilitățile și infrastructura din jur, precum și tendintele pieței. Este esențial ca aceste evaluări să fie cât mai precise și obiective pentru a reflecta valoarea reală a proprietăților și pentru a asigura tranzacții echitabile între cumpărători și vânzători.

În trecut, estimarea prețurilor imobiliarelor s-a bazat în mare parte pe experiența agenților imobiliari sau a evaluatorilor specializați. Cu toate acestea, evoluția tehnologică și dezvoltarea inteligenței artificiale au oferit noi oportunităti și solutii în domeniul imobiliar.

Implementarea acestui proiect se concentrează pe utilizarea inteligenței artificiale pentru a dezvolta o aplicație în domeniul imobiliar prin care utilizatorilor li se vor furniza prețuri estimate ale proprietăților imobiliare, dar și alte informații relevante și actuale despre piața imobiliară.

Astfel, acest proiect reprezintă o soluție inovatoare în domeniul imobiliar, oferind beneficii semnificative pentru ambele parți implicate în tranzacțiile imobiliare și contribuind la imbunatățirea experienței utilizatorilor în acest sector dinamic ajutându-i să ia decizii informate și să identifice oportunități avantajoase.

## Motivația alegerii lucrării de licență

Motivația alegerii acestei teme pentru lucrarea de licență se bazează pe importanța și relevanța sectorului imobiliar în economie și viața cotidiană, tranzacțiile imobiliare reprezentând adesea cele mai semnificative achiziții financiare pe care indivizii le fac în viața lor, fie că este vorba despre achiziționarea unei locuințe sau investiții in proprietăți comerciale.

Dezvoltarea unei aplicații care poate estima prețurile imobilelor aduce multiple beneficii atât pentru cumpărători cât și pentru vânzători. Pentru cumpărători, o astfel de aplicație oferă o sursă de informații obiective și ajută la evaluarea corectă a valorii unei proprietăți în funcție de caracteristicile sale, aceștia având posibilitatea să ia decizii informate și să negocieze in mod eficient prețul. Pentru vânzători, aplicația poate reprezenta un instrument de evaluare a prețului corect al proprietății lor, oferindu-le posibilitatea de a obține un profit adecvat și de a-și maximiza rezultatele financiare.

Această aplicație poate servi și ca un instrument de sprijin în realizarea evaluărilor pentru agenții imobiliari și experții în evaluarea proprietăților. Aceștia pot utiliza datele și predicțiile generate de aplicație pentru a obține o perspectivă mai cuprinzătoare asupra pieței imobiliare și pentru a oferi clienților estimări mai precise și credibile ale valorii proprietăților.

In concluzie, alegerea acestei teme pentru dezvoltarea unei aplicații de estimare a prețurilor imobilelor este motivată de dorința de a aprofunda cunostințele în domeniul inteligenței artificiale și de a cerceta domeniul imobiliar. Prin această inițiativă, contribuim la transformarea digitală a pieței imobiliare și obținem competențe valoroase pentru viitoarele noastre cariere în acest domeniu.

## Structura lucrării

Documentul de față este structurat în opt capitole ample, fiecare abordând diferite aspecte importante legate de proiectul descris anterior și fiind împărțit în mai multe subcapitole.

În primul capitol, se prezintă și se descrie contextul actual în ceea ce privește partea tehnologică a domeniului imobiliar, precum motivația alegerii acestei teme și se realizează o scurtă introducere în ramura specifică a tehnologiei aplicate în domeniul imobiliar.

În al doilea capitol, se definesc obiectivele pe care sistemul prezentat își propune să le atingă în cadrul proiectului, precum și descrierea modului în care aceste obiective vor fi realizate. Totodată, se prezintă tipurile de utilizatori și cerințele funcționale și non-funcționale ale sistemului dezvoltat.

În cadrul capitolului trei, se efectuează o parcurgere a resurselor și a lucrărilor de specialitate realizate anterior, complementare temei abordate în domeniul imobiliar. Acest capitol oferă o imagine de ansamblu asupra cercetărilor și rezultatelor obținute în domeniul tehnologiilor aplicate în piața imobiliară.

În al patrulea capitol, se descrie designul general al soluției propuse, prezentând conceptele teoretice implementate și fundamentul teoretic care susține proiectul. De asemenea, se dezvoltă aspectele abstracte și teoretice relevante pentru înțelegerea arhitecturii software a proiectului.

În capitolul cinci, se documentează detaliile și implementarea aplicației în cadrul proiectului, asigurând astfel dezvoltarea și întreținerea ulterioare a acestuia. Sunt prezentate scheme reprezentative ale aplicației, precum arhitectura folosită și componentele implicate în cadrul sistemului dezvoltat.

Capitolul șase se concentrează pe partea de testare și validare a sistemului dezvoltat în cadrul proiectului, punând în evidență rezultatele obținute și relevanța acestora în contextul problemei propuse în piața imobiliară.

În al șaptelea capitol, se oferă un ghid de utilizare a sistemului dezvoltat pentru utilizatori, împreună cu procesul de instalare al aplicației. Se prezintă cerințele hardware și software necesare pentru o utilizare eficientă a sistemului.

Ultimul capitol al lucrării conține concluziile extrase din studiul prezentat, oferind, totodată, sugestii pentru posibile dezvoltări ulterioare ale aplicației. Aceasta permite sintetizarea rezultatelor și aportului personal adus în domeniul imobiliar, evidențiind contribuțiile și inovațiile aduse de proiect.

# Obiectivele proiectului

Tema proiectului constă în dezvoltarea unui model de predicție a prețurilor proprietăților imobiliare, care va fi integrat și gestionat într-o aplicație web complexă în care administratorul actualizează datele și reantrenează modelul, iar utilizatorii beneficiază de evaluarea automată a proprietăților pe baza unor caracteristici specifice. Scopul principal al proiectului este de a furniza o aplicație suport pentru procesul decizional privind achiziționarea sau vânzarea unei proprietăți. Pentru a atinge acest scop, trebuie îndeplinite următoarele obiective:

## Obiective principale

1. *Dezvoltarea unui model de predicție a prețului*

Unul dintre obiectivele principale ale proiectului este crearea unui model de inteligență artificială care să efectueze predicții precise ale prețurilor proprietăților imobiliare. Acest model va fi antrenat pe un set de date extins, care conține caracteristici relevante pentru evaluarea prețurilor, cat si pe un set de date actuale colectate manual de pe Internet. Modelul va fi proiectat astfel încât să utilizeze algoritmul regresiei liniare pentru efectuarea predicției. Scopul final este de a obține un model cu o acuratețe cat mai buna pentru a furniza utilizatorilor estimări cât mai precise.

1. *Colectarea de date actuale de antrenament*

Un alt obiectiv principal este colectarea si actualizarea setului de date folosit pentru antrenarea modelului. Pentru a obține predicții mai precise, este necesară extragerea datelor actuale de antrenament de pe diverse site-uri de anunțuri imobiliare. Aceste date actualizate vor contribui la îmbunătățirea acurateței predicțiilor și la reflectarea tendințelor actuale din piața imobiliară.

1. *Implementarea funcționalității de estimare a prețului*

O altă componentă importantă a proiectului este implementarea unei funcționalități care să permită utilizatorilor să introducă caracteristicile propriilor proprietăți și să primească o estimare a prețului acestora. Aceasta functionalitate va folosi modelul construit si va oferi utilizatorilor informațiile necesare pentru a evalua valoarea proprietăților lor și a lua decizii informate în privința tranzacțiilor imobiliare.

1. *Actualizarea constantă a datelor*

Pentru a menține precizia predicțiilor, este esențială actualizarea constantă a datelor folosite în cadrul aplicației. Aceasta implică monitorizarea și colectarea permanentă a informațiilor legate de prețurile și caracteristicile proprietăților imobiliare, astfel încât utilizatorii să beneficieze de estimări actualizate și relevante. Aceast lucru se poate face manual de catre o persoana autorizata, si la extragerea anunturilor se va tine cont de data de publicare pentru a asigura ca anunturile sunt actuale si nu exista deja in setul de date.

## Obiective secundare

1. *Realizarea unui sistem de înregistrare și autentificare*

Utilizatorii aplicației vor putea să iși înregistreze un cont în aplicație și să se autentifice. De asemenea, se vor putea diferentia utilizatorii dupa rol in momentul autentificarii, ceea ce va permite limitarea accesului utilizatoriilor de tip client la anumite actiuni atribuite administratorului aplicatiei. Pentru început utilizatorii care sunt autentificați vor avea, in plus fata de cei ce nu sunt, capacitatea de a vizualiza statisticile care sunt prezentate in obiectivul secundar 3. În vreme ce, administratorul va avea acces la toate paginile si functionalitățile aplicației.

1. *Vizualizarea și filtrarea anunțurilor imobiliare*

Utilizatorii vor avea posibilitatea de a vizualiza anunțurile imobiliare actuale și de a naviga către pagina care conține anunțul original pentru mai multe informații. În plus, acestia vor putea aplica diferite filtre asupra anunțurilor pentru a le vizualiza strict doar pe cele care îndeplinesc caracteristicile căutate.

1. *Vizualizarea statisticilor relevante*

Administratorul și utilizatorii care dețin un cont și sunt autentificați vor putea să vizualizeze diferite reprezentări grafice cu referire la diverse aspecte ale pieței imobiliare și grafice care reflectă acuratețea modelui de predicție. Această funcționalitate va oferi utilizatorilor o perspectivă mai amplă asupra pieței imobiliare, dar ii va și ajuta să iși dea seama cât de bună este predicția făcută de model.

1. *Implementare unei interfete intuitive*

Unul dintre cele mai importante obiective secundare este implementarea unei interfețe bine structurate, aceasta trebuie să fie cât mai intuitivă pentru a oferi o experiență prietenoasă utilizatorului. In urma definirii obiectivelor anterioare , interfata utilizator va trebui să conțină următoarele cate o pagina pentru fiecare dintre următoarele funcționalități: înregistrare, logarea în aplicție, vizualizarea și filtrarea anunțurilor, estimarea prețurilor, vizualizarea statisticilor si administrator.

# Studiu bibliografic

## Test

Documentarea bibliografică are ca obiectiv prezentarea stadiului actual al domeniului/sub-domeniului în care se situează tema. În redactarea acestui capitol (în general a întregului document) se va ține cont de cunoștințele acumulate la disciplinele dedicate din semestrul 2, anul 4 (Metodologia Întocmirii Proiectelor, etc.), precum si la celelalte discipline relevante temei abordate.

Acest capitol va avea între 3 și 10 pagini.

Referințele se scriu în secțiunea *Bibliografie*. Formatul referințelor trebuie sa fie de tipul *IEEE* sau asemănător. Introducerea și formatarea referințelor în bibliografie, respectiv citarea în text, se poate face manual – nu recomandăm, întrucât sunt mai greu de gestionat corect – sau folosind instrumentele de lucru menționate în ultimele paragrafe din acest capitol.

In secțiunea *Bibliografie* sunt exemple de referințe pentru articol la conferințe [1] [2], articol în jurnal [3], sau cărți [4]. Referințele spre aplicații sau resurse online (pagini de internet) trebuie sa includă cel puțin o denumire sugestivă pe lângă URL ca în [5], plus alte informații dacă sunt disponibile (autori, an, etc.) și data ultimului acces. Referințele care prezintă doar o legătură spre resursa online se vor plasa în footer-ul paginii unde sunt referite.

Citarea referințelor în text este obligatorie, vezi exemplul de mai jos (în funcție de tema proiectului se poate varia modul de prezentare a metodei/aplicației).

În articolul [1] autorii prezintă un sistem pentru detecția obstacolelor în mișcare folosind stereo viziune și estimarea mișcării proprii (nu e adevărat despre articolul [1] ). Metoda se bazează pe …*trecerea în revistă a algoritmilor, structurilor de date, funcționalitate, aspecte specifice temei proiectului* *etc*….. Discuție: *avantaje – dezavantaje*.

În capitolul 4 al [4] *se prezintă* …..

Începând cu **MS Word 2007** se poate folosi sistemul integrat de gestiune bibliografiei: *References* submeniul *Citations & Bibliography.* Mai multe informații se găsesc în documentația online de la MS Office.

# Analiză și fundamentare Teoretică

## Modelul de predicție a prețului

### Seturile de date utilizate

Sistemul creat lucreaza cu doua seturi de date, un set de date mare si cu anunturi mai vechi si un set de date actual construit manual. Setul de date folosit are o foarte mare importanta in dezvoltarea si evaluarea unui model inteligent.

Primul set de date [6] contine caracteristicile unor proprietati de vanzare din Bucuresti. Cu toate ca aceste date vechi furnizeaza doar o perspectiva istorica asupra preturilor, datorita faptului ca setul de date este destul de mare, continand pana la 8321 de anunturi putem obtine un model mult mai bun de predictie a pretului, chiar daca acesta nu este in concordanta cu preturile pietei actuale.

Al doilea set de date este creat manual prin colectarea de anunturi imobiliare din Cluj-Napoca de pe site-ul Blitz. Acest set de date are dimensiuni mai mici, dar ne ajuta sa oferim o imagine mai actuala a pietei imobiliare din aceasta zona.

### Preprocesarea datelor

Preprocesarea datelor este un pas foarte important in antrenarea modelelor de predictie. Astfel, ne vom asigura ca fom folosi datele cele mai relevante pentru a prezi pretul.

Primul pas este acela de a ne asigura ca datele colectate sunt relevante pentru analiza pretului, ca provin din surse sigure si ca sunt suficient de multe pentru o prezicere cat mai buna.

Al doilea pas presupune identificarea si tratarea valorilor lipsa si eliminarea duplicatelor. Putem alege sa eliminam inregistrarile cu date lipsa, sa inlocuim aceste valori cu medii, mediane sau sa folosim tehnici avansate precum imputarea datelor lipsa.

A screenshot of a computer

Description automatically generated

Pentru acest pas folosim functii speciale pentru analiza datelor, iar in cazul coloanelor irelevante care au foarte multe valori lipsa cum este in cazul nostru coloana “Facilitati” le vom elimina, iar pentru coloanele relevante cu valori null vom atribui valoarea 0 pentru acestea cum ar fi coloana “Nr. Balcoane”.

Un alt pas in preprocesarea datelor il constituie codificarea valorilor categorice. In cazul in care avem variabile categorice precum “cartier”, “boxa subsol”, adica variabile care influenteaza valoarea pretului dar nu au valori numerice, trebuie sa le codificam intr-o forma numerica pentru a putea fi folosite in regresia liniara.

In urma analizei vizuale si nu numai vom transpune valori precum “Garsonira” pentru coloana “nr. Camere” in valoarea 1 si pentru coloana “Boxa subsol” valorii “Nu” ii vom atribui 0, iar valorii “Da” ii vom atribui 1.

A screenshot of a table

Description automatically generated

Ultimul pas il constituie separarea datelor in setul de antrenare si celd e testarea. Setul de antrenare va fi folosit pentru construirea modelului, in timp ce setul de testare va fi folosit pentru a evalua performanta acestuia.

### Algoritmi de codificare a locațiilor

In domeniul imobiliar , zona in care este situata o proprietate dintr-un anunt are un impact foarte mare asupra pretului, de aceea avem nevoie de un algoritm cat mai bun pentru a include si coloana destinata locatiei sub forma unui cod in modelul de predictie.

Un prim algoritm care a fost incercat se bazeaza pe atribuirea codului unei locatii pe baza ordonarii crescatoare a locatiilor in functie de pret. Mai intai se grupeaza toate anunturile dupa locatie si se realizeaza media pretului pe unitatea de suprafata pentru fiecare locatie. Dupa calcularea mediilor se ordoneaza locatiile crescator in functie de pretul mediu si li se vor atribui coduri de la 1 pentru locatia cu cel mai mic pret mediu pana la numarul total de locatii pentru locatia cu cel mai mare pret mediu.

Al doilea algoritm incepe tot cu gruparea inregistrarilor dupa locatie si calcularea pretului mediu pe unitatea de suprafata pentru fiecare locatie. Apoi se va calcula media pretului mediu pe fiecare locatie si se va folosi in calcularea codului impartind valoarea medie pe locatie la media generala. Prin utilizarea acestui algoritm de codificare, obținem o reprezentare numerică a locației care conține informații sugestive despre costul imobiliar în diferitele cartiere.

Deși primul algoritm poate fi mai simplu în comparație cu al doilea, in construirea modelului l-am ales pe cel de al doilea, deoarece are avantajul că păstrează informații despre relația de ordine dintre prețurile medii ale locațiilor. Deci, într-un model de regresie liniară, algoritmul poate ajuta la capturarea variațiilor de preț și a subtilităților între locații, ceea ce poate duce la predicții mai precise ale prețurilor imobiliare.

### Utilizarea regresiei liniare pentru predictia pretului

Regresia liniară este o tehnică de modelare statistică utilizată pentru a prezice valoarea unei variabile pe baza uneia sau mai multor variabile. Variabila care se doreste sa fie prezisa se numeste variabila dependenta, iar celelalte variabile utilizate in predictie se numesc variabile independente. Aceasta metoda estimeaza coeficientii ecuatiei liniare, implicand una sau mai multe variabile independente. Coeficienții din ecuația de regresie indică direcția și magnitudinea impactului fiecărei variabile independente asupra variabilei dependente.

Ecuația unui model de regresie liniară simplă este dată de:

**Y = β0 + β1X + ε**,

unde Y este variabila dependentă, X este variabila independentă, β0 și β1 sunt coeficienții de regresie, iar ε este termenul de eroare.

Regresia liniara este o alegere buna pentru implementarea modelului de predictie a pretului, deoarece ofera o formula matematica usor de interpretat si asigura antrenarea rapida a modelului. Într-o aplicație specifică în industria imobiliară, regresia liniară este utilizată pentru a realiza predicții privind prețurile proprietăților. Aici, variabila dependentă este prețul proprietății, iar variabilele independente sunt reprezentate de diverse caracteristici ale proprietății care ar putea influența prețul.

Pentru setul de date din Cluj-Napoca, variabilele independente sunt urmatoarele: numărul de camere, suprafața, numărul de băi, numărul de balcoane, etajul, numărul de locuri de parcare, existența unei boxe în subsol, codificarea cartierului și vechimea clădirii. Ecuatia acestui model va arata in felul urmator:

Preț proprietate = β0 + β1 × Nr. Camere + β2×Suprafața +…+ ε

Coeficienții reflectă impactul fiecărei variabile independente asupra prețului proprietății. De exemplu, coeficientul asociat numărului de camere ar putea arăta cât de mult crește prețul pe măsură ce numărul de camere crește. Prin ajustarea coeficienților în funcție de datele disponibile, modelul poate prezice prețurile estimative ale proprietăților pe baza caracteristicilor lor.

### Utilizarea Random Forest pentru predictia pretului

Random Forest (Pădurea Aleatoare) este o tehnică de învățare automată utilizată pentru atât probleme de regresie, cât și de clasificare. Acesta este un tip de algoritm ensemble, ceea ce înseamnă că combină mai multe modele mai simple pentru a obține un rezultat mai robust și mai precis.

Random Forest este compus din mai multe arbori de decizie. Fiecare arbore de decizie este construit pe un set diferit de date, ales aleatoriu din setul de antrenare. Fiecare arbore de decizie produce o predicție independentă, iar rezultatul final al Random Forest este o medie (în cazul regresiei) sau o majoritate (în cazul clasificării) a predicțiilor individuale ale arborilor. O caracteristică cheie a Random Forest este că poate gestiona atât caracteristicile continue, cât și cele categorice fără a necesita preprocesare extensivă.

În cazul predicției prețurilor în imobiliare, Random Forest poate fi folosit în mod similar cu regresia liniară pentru a estima prețurile proprietăților. Fiecare arbore de decizie din Random Forest va lua în considerare diverse caracteristici ale proprietăților și va prezice prețurile individuale, iar rezultatul final va fi o medie a acestor predicții.

Caracteristici ale proprietăților (variabile independente) precum numărul de camere, suprafața utilă, numărul de băi, etajul, existența balcoanelor, vechimea clădirii etc. pot fi folosite pentru construirea Random Forest. Coeficienții din ecuația regresiei liniare sunt înlocuiți aici de importanța caracteristicilor determinate de Random Forest.

Astfel, Random Forest poate să ofere o predictibilitate mai bună decât un singur model de regresie liniară, deoarece abordează mai bine variabilitatea din date prin utilizarea unor multiple modele.

### Utilizarea clasificatorului Support Vector Machines pentru predictia pretului

Support Vector Regression (SVR) este o variantă a algoritmului Support Vector Machine (SVM) adaptată pentru probleme de regresie. În loc să încerce să separare clasele, cum face SVM în clasificare, SVR încearcă să găsească o linie (pentru regresie liniară) sau o curbă (pentru regresie neliniară) care se potrivește cel mai bine cu datele de antrenare și minimizează erorile de predicție.

SVR implică găsirea unei linii sau a unei curbe care maximizează distanța (marginea) dintre punctele de antrenare și linia/curba de regresie. Marginea este definită de două puncte de suport, care sunt cele mai apropiate puncte de linie/curbă. Aceste puncte de suport sunt cele care influențează direct construirea liniei/curbei de regresie.

### Algoritmul de detectare a anomaliilor

Detecția anomaliilor este procesul de identificare a punctelor de date sau a observațiilor care se abat semnificativ de la comportamentul general sau normal al unui set de date. Aceste puncte, numite "anomalii" pot fi cauzate de erori ale datelor, evenimente rare sau neașteptate, sau pot semnala situații de interes în sine. Acest proces are scopul de a identifica aceste anomalii pentru a putea investiga și lua măsuri corespunzătoare în funcție de contextul problemei.

Algoritmii de detecție a anomaliilor implică în general următorii pași:

* **Definirea normalității**: Este necesar să se stabilească ce este considerat "normal" în setul de date. Acest lucru poate fi realizat prin analiza distribuției datelor sau a comportamentului obișnuit.
* **Alegerea metodei**: Există numeroase metode de detectare a anomaliilor, cum ar fi bazate pe statistici, distanțe, modele matematice sau modele de învățare automată. Fiecare metodă are propriile sale avantaje și limitări.
* **Calcularea deviației**: Algoritmul măsoară cât de mult se abate fiecare punct de date de la normalitate. Această deviație poate fi calculată folosind metrici precum distanțe euclidiene, diferențe de medie, deviație standard sau alți indicatori specifici.
* **Stabilirea pragului**: Un prag este stabilit pentru a delimita între anomaliile și datele normale. Punctele care depășesc acest prag sunt considerate a fi anomalii.
* **Identificarea anomaliilor**: Punctele de date care trec peste pragul sunt identificate și marcate ca anomalii. Acestea pot fi considerate pentru ulterioră investigație.
* **Interpretarea rezultatelor**: Rezultatele obținute sunt analizate pentru a înțelege natura și cauza anomaliilor. Uneori, o anomalie poate fi justificată de circumstanțe speciale sau erori în date.
* **Acțiuni următoare**: În funcție de context, se pot lua măsuri pentru a rezolva sau investiga cauzele anomaliei. Aceste măsuri pot varia de la simpla corecție a datelor până la acțiuni precum stergerea acetsora.

## Colectarea datelor de pe internet

În era informației digitale, colectarea datelor de pe internet a devenit esențială pentru cercetători, analiști și dezvoltatori de software. O metodă eficientă de extragere a datelor din paginile web este data scraping, o tehnică care implică extragerea automată a datelor din paginile web pentru analiză ulterioară. Una dintre cele mai utilizate abordări în data scraping implică utilizarea driverului Chrome.

Am ales să colectăm date de pe site-ul "Blitz Imobiliare". Acest site este cunoscut pentru a oferi informații detaliate despre proprietăți disponibile, inclusiv caracteristici precum numărul de camere, suprafață, etaj, cartier, preț etc. De asemenea, site-ul se actualizează frecvent, asigurând astfel date relevante pentru analiză.

Prin alegerea unei surse precum "Blitz Imobiliare", ne asigurăm că avem acces la date de calitate și actualizate, oferindu-ne o bază solidă pentru analiză. Utilizând tehnica de data scraping cu driverul Chrome, putem colecta eficient datele relevante și le putem transforma în informații valoroase pentru analiza noastră în contextul regresiei liniare aplicate pe piața imobiliară.

Pasii pentru procesul de colcetare al anunturilor sunt urmatorii:

* Instalarea Driverului Chrome: Descarcă și instalează driverul Chrome corespunzător versiunii browserului tău Chrome.
* Selectarea Elementelor: Identifică elementele HTML pe care dorești să le extragi din pagina web. Acestea pot fi selecționate folosind selecția CSS sau XPath.
* Configurarea în Python: Utilizează o bibliotecă precum Selenium în Python pentru a interacționa cu driverul Chrome. Încarcă pagina, găsește elementele dorite și extrage-le.
* Preluarea Datelor: După identificarea elementelor, preia valorile lor (text, linkuri, imagini etc.) și transformă-le în formatul dorit (CSV, JSON etc.).
* Iterație și Navigare: Pentru paginile web cu mai multe pagini sau cu conținut generat dinamic, va trebui să iterezi prin pagini și să simulezi acțiuni precum scroll sau click pentru a prelua toate datele dorite.

## Structura logică a aplicației

## Medii de programare

Un aspect esential în realizarea aplicatiei il constituie alegerea limbajului de programare, deoarece fiecare limbaj e diferit si ofera diferite lucruri si acesta sta la bazele înțelegerii cat mai rapide a proiectului. Astfel, alegerile facute au fost unele strategice in ceea ce priveste toate componentele aplicației. Pentru partea de inteligenta artificiala si web scraping, am optat pentru Python, un limbaj recunoscut pentru versatilitate si eficienta in acest domeniu. Totodata, Python vine cu o gama larga de librarii specializate si o sintaxa concisa, cee ace faciliteaza implementarea de algoritmi complecsi si exploatarea solutiilor de invatare automata.

În ceea ce privește gestionarea utilizatorilor, am ales sa folosesc Java in combinație cu framework-ul Spring. Java este cunoscut pentru fiabilitate si securitatea sa, iar Spring furnizeaza un cadru de lucru solid pentru dezvoltarea aplicatiilor web scalabile si robuste. Aceasta alegere asigura o gestionare eficienta a utilizatorilor si un mediu sigur pentru interactiunea cu platforma.

Pentru partea de frontend, am ales sa folosesc limbajul JavaScript impreuna cu biblioteca React. JavaScript este un limbaj de programare destinate dezvoltarilor de interfete utilizator, iar acesta folosit impreuna cu React ofera o dezvoltare modulara si eficienta pentru construirea interfetelor web creand astfel experiente de utilizare intuitive. Tot la partea de frontend am folosit CSS si SCSS pentru stilizarea interfetei.

In ceea ce priveste partea de server a componentei inteligente, am utilizat framework-ul Flask in Python. Flask ofera un mediu usor de utilizat si flexibil pentru gestionarea serviciilor web.

Fiecare limbaj si tehnologie selectate au fost evaluate minutios pentru a se potrivi cerintelor fiecarei componente in parte si pentru a asigura o dezvoltare si intretinere usoara pe termen lung.

# Proiectare de detaliu și implementare

In capitolul acesta, vom documenta detaliat aplicatia dezvoltata cu scopul de a contribui la o dezvoltare ulterioara si intrentinere mai eficienta. Viitorii dezvoltatori vor putea identifica si intelege modulele principale ale aplicatiei mai usor . Se vor prezenta

schema generala a aplicatie, diagrame de clase si implementarea metodelor importate pentru a concretiza modul de functionare al aplicatiei. De asemenea, se vor explica si alegerile tehnologice impreuna cu motivele din spatele acestora.

## Arhitectura generală a aplicației

Solutia implementata pentru acest proiect reprezinta un sistem compus din trei aplicatii, fiecare avand responsabilitatii specifice si contribuind la functionarea generala a aplicatiei. Scopul alegerii unei arhitecturi modulare a fost acela de a separa functionalitatile aplicatiei asigurand astfel o dezvoltare ulterioara si intretinere mult mai optima, cat si pentru a acoperii cat mai multe cunostinte acumulate .

Prima aplicatie se ocupa cu implemntarea interfetei web reprezentand partea vizibila a aplicatiei. Celelalte doua functioneaza ca servere si au roluri specifice in cadrul sistemului.

Aplicatia dezvoltata in Java se ocupa cu gestionarea utilizatorilor, mai exact cu implementarea functionalitatilor de inregistrare si autentificare. Aceasta stocheaza datele despre utilizatori intr-o baza de date MySql.

Aplicatia dezvoltata in Python reprezinta componenta inteligenta a sistemului, ea fiind responsabila cu predictia preturilor si colectarea anunturilor de pe Internet.

Aceste trei aplicații funcționează într-un mod integrat și interacționează între ele pentru a oferi o experiență completă și funcțională utilizatorilor. Astfel, prin separarea responsabilităților și utilizarea tehnologiilor și limbajelor de programare adecvate, această arhitectură permite o dezvoltare modulară și scalabilă a aplicației,

A diagram of a computer program

Description automatically generated

## Structura programului

### Diagrame de pachete

Diagrama de pachete pentru aplicatia Web este prezentată in Figură 5‑1.

Pachetul hooks este responsabil pentru gestionarea hook-urilor personalizate. Hooks-urile in React sunt niste functii speciale care permit componentelor functionale sa aiba stari interne si sa utilizeze functionalitati specifice clasei, fara a fi nevoie sa fie convertite in componente bazate pe clase. Un hook personalizat, pe de alta parte, este un hook creat de dezvoltatori din mai multe hooks predefinite.

Pachetul constants contine fisierele care contin constante utilizate in intreaga aplicatie. Constantele joaca un rol important in furnizarea de valori fixe care raman consistente in diferitele parti ale codului. Separarea acestora intr-un pachet dedicat, le face mai usor de gestionat si actualizat atunci cand este necesar.

Pachetul pages contine pachete individuale corespunzatoare fiecarei pagini a aplicatiei, in interiorul fiecarui pachet de pagina se afla componenta paginii, care defineste structura si functionalitatea acelei pagini, precum si fisierul pentru stilizare asociat.

Pachetul api contine fisierele responsabile pentru configurarea comunicarii dintre frontend si backend.

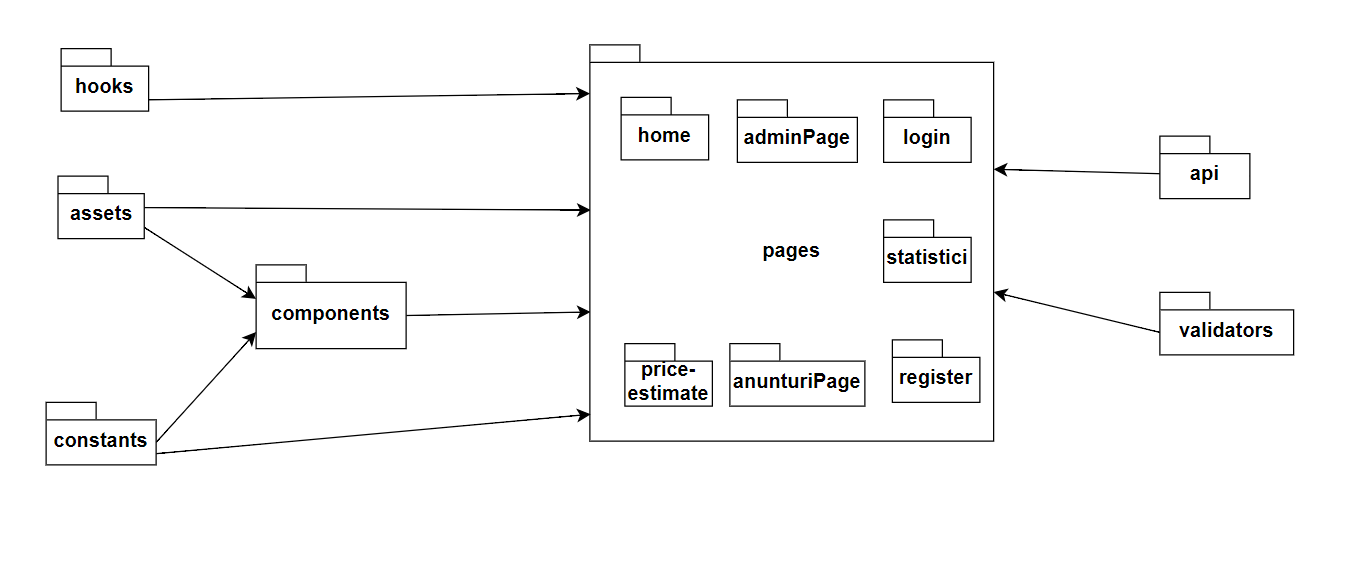
Pachetul assets cuprinde diverse resurse utilizate in cadrul aplicatiei pentru a oferi continut vizual, acestea pot fii: imagini, fonturi, videoclipuri, sunete, etc.

Pachetul components cuprinde diverse componente reutilizabile si de uz general

in celelalte componente ale aplicatiei.

Pachetul validators contine diversi validatori responsabili cu validrea datelor introduse de utilizator. Acești validatori efectuează verificări și aplică reguli sau constrângeri specifice asupra introducerii utilizatorului, asigurând integritatea și acuratețea datelor.

Toate aceste pachete se afla in pachetul src, care reprezinta pachetul radacina si include componenta principala a aplicatiei.



Figură 5‑1: Diagrama de pachete a aplicației Web

*Diagrama de pachete pentru aplicatia de gestiune a utilizatorilor*

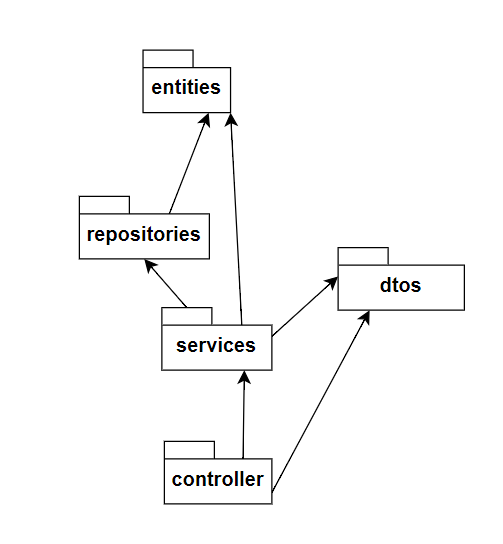
Pachetul PricePrediction este pachetul radacina si cuprinde totalitatea pachetelor si clasa principala a aplicatie.

Pachetul entities contine clasele destinate reprezentarii obiectelor care vor fi stocate in baza de date.

Pachetul repositores contine clasele care implementeaza operatiile pe baza de date.

Pachetul service contine clasele care implementeaza functionalitatiile logice ale aplicatiei.

Pachetul controller contine controllerul aplicatiei , cel care implementeaza metodele care vor fi apelate atunci cand frontend-ul va trimite o cerere.



### Diagrame de clase

A screenshot of a computer

Description automatically generated

## Modulul de predicție a prețului

### Clasa loadData

Aceasta clasa contine o metoda care primeste ca si parametru un path catre un fisier excel de unde va citii datele si le va returna.

### Clasa Preprocesare

Acest cod definește o funcție numită preprocesare care primește un DataFrame data și aplică unele operații de preprocesare asupra acestuia:

### Clasa de codificare a locatiilor

### Clasa pentru antrenarea modelului

### Clasa pentru calcularea perfomantei modelului

### Clasa pentru detectia si eliminarea anomaliilor

### Clasa pentru generarea graficelor

## Modulul de colectare a datelor de pe internet

Functia de colectare a datelor de pe Internet a fost proiectata in asa fel incat sa se poata colecta atat anunturi pentru Bucuresti , cat si anunturi pentru Cluj.

Functia primeste ca parametrul orasul pentru care se doreste colectarea. Acest parametru este transmis prin ruta din frontend cu numele de city si de aceea se apeleaza functia de get a argumentelor pentru a lua valoarea acestuia. In functie de oras vom initializa variabilele care au valori diferite pentru cele doua orase, acestea sunt: existing\_data – datele deja existente in excel, output\_filename – numele fisierului in care se vor adauga datele colectate, url – url-ul care se acceseaza pentru colectarea anunturilor si zona- denumirea tagului care indica locatia din care este anuntul.

Dupa aceste atribuiri, vom extrage cea mai recenta data de publicare a unui anunt care se afla deja in setul de date, utilizand functia max().

In variabila PATH de declara calea catre fisierul executabil al driverului Chrome utilizat pentru web scraping. Apoi in variabila driver se va creea o instanta a driverului Chrome utilizand calea specificata in variabila PATH.

Prin intermediul comenzii driver.get(url) se va accesa pagina aferenta link-ului trimis ca parametru.

Pentru a evita anumite restrictii aplicate de site vom folosi variabila count pentru a contoriza cate anunturi au fost colectate.

In urma analizei codului sursa al site-ului cu anunturi am observat ca avem o lista initiala cu toate anunturile si fiecare anunt se regaseste in elemente care au clasa card, prin urmare extragem toate aceste elemenete cu functia find\_elements si le stocam in variabila cards

## Aplicația Web

### Comunicarea între Backend si Frontend

Un aspect fundamental în dezvoltarea aplicațiilor web îl reprezintă comunicarea dintre Frontend și Backend. În cadrul acestui proiect, comunicare se realizează prin intermediul protocolului de transfer de date HTTP. Pentru schimbul de informații între client și server, Frontend-ul poate face cereri HTTP, Backend-ul primește cererile și le procesează iar după procesare transmite un raspuns HTTP către frontend care conține informațiile solicitate sau un status care indica dacă cererea s-a executat cu succes sau a intervenit o eroare.

În fișierul **httpConfig.js** se configurează și se creează instante Axios pentru efectuarea cererilor HTTP.

Mai întai se definesc adresele de baza pentru servere ținând cont de portul pe care acestea rulează:

* ‘**javaEndpoint**’ reprezintă adresa de baza pentru serverul Java care rulează pe portul 8085 (http://localhost:8085)
* ‘**pythonEndpoint**’ reprezintă adresa de bază pentru serverul Python care rulează pe portul 5000 (<http://localhost:5000>)

Apoi se creează doua instanțe separate de obiecte Axios ‘javaHttp’ și ‘pythonHttp’ pentru fiecare server utilizând metoda ‘axios.create()’, aceste instanțe sunt configurate pentru a efectua cereri HTTP către două servere distincte, metoda create() are doi parametrii: baseUrl reprezintă URL-ul de bază adică adresele definite anterior iar al doilea parametru, headers e folosit pentru a specifica formatul de date care se acceptă și tipul de conținut trimis către server, în cazul nostru configurăm cererile ca să trimită și să accepte date cu format JSON.

Istanțele de obiecte Axios sunt exportate și folosite ulterior în fișierul httpUtils la construirea unor funcții generice care au ca parametrii un URL și alți parametrii opționali și care vor fi folosite în intreaga aplicație pentru realizarea cererilor HTTP de tip GET, POST, PUT, PATCH si DELETE către cele două servere.

Aceste funcții sunt denumite sugestiv pentru a putea face diferența dintre serverul către care se va face cererea și scopul lor este de a abstractiza detalii precum gestionarea anteturilor și a tipului de conținut.

### Componenta de meniu

Componenta **Menu** reprezintă bara de meniu a aplicației, aceasta asigura vizibilitatea elementelor de meniu in functie de diferite conditii precum tipul de utilizator si starea de logare a acestuia.

La inceputul componentei se extrag variabilele de stare de tip bool care ne indica daca utilizatorul este logat (isLoggedIn) respectiv daca acesta este administratorul aplicatiei (isAdmin). Pentru ca aceste variabile trebuie sa aiba o valoare persistenta in toate componentele aplicatiei, acestea s-au definit in contextul componentei principale (AppContext) , iar pentru a le putea accesa si modifia in componenta Menu se foloseste hook-ul useContext pentru a accesa variabilele din contextul principal.

Functia onClickLogout din aceasta componenta este apelata cand utilizatorul se delogheaza, respectiv apasa pe optiunea Logout din meniu. Aceasta functie seteaza variabila isLoggedIn si isAdmin la false, efectuand astfel delogarea.

Partea de return a componentei cuprinde ceea ce va fi randat in aplicatie, mai exact in tag-ul header se afla toate componentele meniului, care sunt afisate in functie de starea de autentificare si tipul utilizatorului folosindu-se randearea conditionata.

Daca utilizatorul normal sau de tip administrator nu este logat, in meniu vor aparea doar optiunile Login, Register, Anunturi, Evaluarea proprietate si Logo-ul aplicatiei, care reprezinta link-ul catre pagina principala a aplicatiei.

Daca utilizatorul este logat si nu este de tip admin atunci pentru acesta vor aparea urmatoarele optiuni: Logo-ul, Anunturi, Statistici, Evaluare proprietate, si Logout. Iar daca utilizatorul este de tip admin si este logat, pe langa cele mentionate anterior ii va aparea optiunea Admin.

Fiecare optiune din meniu este reprezentata prin intermediul componentelor NavItem din biblioteca reactstrap , respectiv Link din biblioteca react-router-dom, in interiorul acestor tag-uri se trece continutul care dorim sa fie afisat ca optiune in meniu iar in tag-ul de Link la atributul to se va pune URL-ul catre care se va naviga in comentul apasarii optiunii.

### Componenta de login

O componenta importanta a aplicatiei o reprezinta componenta de Login prin intermediul careia utilizatorul in urma completarii datelor necesare se poate autentifica in contul sau.

Starile interne principale ale componentei sunt:

* email – stare folosita pentru stocarea valorii emailului introdus de utilizator in form
* password – stare folosita pentru stocarea parolei introduse de utilizator in form
* wrongDataMessage – este o variabila de stare care contine mesajul de eroare care va fi afisat daca utilizatorul introduce credentiale gresite.
* navigate – este o variabila care ne ajuta sa redirectionam utilizatorul catre o alta pagina in urma unei actiuni

Functiile onEmailChanged si onPasswordChanged primesc ca si parametru valoarea introdusa de utilizator in campul email, respectiv password si sunt apelate de fiecare data cand valoarea acestora se schimba, in interiorul acestor functii se seteaza variabilele de stare corespunzatoare.

Functia onLoginClicked se apeleaza in momentul in care utilizatorula apasa butonul de Login. Aceasta functie este una asincrona datorita cererii de tip POST trimisa catre backend la ruta /login, ca si parametru al cererii vom trimite un obiect format din credentialele introduse de utilizator. Daca rezultatul returnat este diferit de null atunci verificam care este rolul utilizatorului, in cazul in care acesta este de tip admin vom seta variabila de context isAdmin cu valoarea true, iar apoi indiferent de tipul utilizatorului vom seta variabila isLoggedIn la tru si vom naviga catre pagina de home a aplicatiei. Daca in baza de date nu este gasit un utilizator cu credentialele respective, cererea va returna un status de eroare si vom afisa un mesaj sugestiv.

Componenta vizuala este reprezentata de un form cu doua input-uri, un buton de Login care va fi activ doar in momentul in care sunt introduse valori in ambele campuri, mesajul de eroare in cazul unei autentificari esuate, si un link catre pagina de inregistrare in cazul in care utilizatorul nu are un cont.

### Componenta de Inregistrare cont

Componenta Register permite utilizatorului sa se inregistreze prin completarea unui formular. Prin intermediul acestei componente se pot inregistra doar utilizatori de tip clienti, utilizatorul de tip administrator fiind unul singur adaugat manual in baza de date.

Variabilele de stare ale acestei componente sunt:

* formData – aceasta variabila stocheaza datele introduse de utilizator in formular si are structura unui obiect cu campurile corespunzatoare input-urilor din formular( firstname, lastname, email, password, confirmPassword).
* errorMessage – e o variabila care stocheaza mesajele de eroare legate de toate campurile formularului, pentru campul email avem mesajul de eroare care va aparea atunci cand utilizatorul introduce un format gresit de email, pentru campurile password si confirmPassword avem mesajul de eroare in cazul in care parola nu este suficient de complexa si nu respecta formatul cerut, si un al mesaj de eroare este destinat introducerii gresite a parolei confirmate
* passwordConfirmed, validEmail si registrationSuccess – reprezinta variabile de stare de tip bool care vor avea valoarea true in cazul in care avem parola initiala se potriveste cu cea confirmata, avem un email valid, respectiv inregistrarea s-a efectuat cu succes, iar in caz contrar acestea vor avea valoarea false.

Functia onInputChange este apelata de fiecare data cand valoarea unui camp din formular se schimba si are rolul de a actualiza starea variabilei formData si de a valida valorile introduse.

Functia handleFormSubmit este apelata atunci cand utilizatorul apasa pe butonul Register si are scopul de a trimite catre backend datele introduse de acesta in formular prin intermediul unei cereri de tip POST catre ruta /register. Daca toate campurile au fost valide si cererea s-a realizat cu succes atunci variabila registrationSucces va fi setata la true.

Functia validateInput valideaza valorile introduse de utilizator in formular si seteaza mesajul de eroare specific fiecare validari esuate, si tot in functie de rezultatul validarii seteaza si valoarea starilor passwordConfirmed si validEmail.

Componenta returneaza un formular cu urmatoarele campuri : nume, prenume, email, parola si confimare parola si un buton de Register care va fi inactiv cat timp valorile din campuri nu sunt valide. Mesajele de eroare se vor afisa sub campul carora le corespund , iar daca inregistrarea s-a efectuat cu succes se va afisa un mesaj sugestiv.

### Pagina de vizualizare a anunturilor

Pagina destinata anunturilor ne permite sa le vizualizam caracteristicile, sa navigam catre site-ul de pe care provine anuntul respectuv si sa aplicam diferite filtre pentru a vizualiza doar anunturile care ne intereseaza. Implementarea si reprezentarea vizuala a acestor functionalitati se regasesc in componenta Anunturi.

Starile interne ale acestei componente sunt urmatoarele:

* excelData – aceasta variabila e folosita pentru a stoca lista de anunturi afisata in interfata
* filters – este o variabila de tip obiect care va contine valorile filtrelor dorite, acest obiect contine urmatoarele etichete: nrCamere, compartiment, cartier, mai exact atributele dupa care se poate efectua filtrarea.
* filteredData – aceasta variabila stocheaza anunturile care satisfac filtrele selectate

Dupa declararea variabilelor de stare, am implementat extragerea anunturilor din fisierul de pe disc in interiorul unui useEffect, deoarece pentru a avea versiunea finala a datelor stocate ne dorim ca la fiecare rerandare a paginii de anunturi sa se actualizeze datele in cazul in care acest lucru s-a intamplat in fisierul excel. Pentru extragerea propriu-zisa se realizeaza o cerere de tip GET catre serverul de python cu ruta /read-excel. Daca aceasta cerere s-a executat cu succes rezultatele sunt stocate in starea excelData si se seteaza si starea variabilei filteredData tot cu acest rezultat, iar daca cererea esueaza se va afisa eroarea in consola.

Urmatoarea functie, handleFilterChange se va apela de fiecare data cand se modifica o valoare indiferent de campul in care aceasta se modifica, in interiorul functiei setam valoarea variabilei de stare cu noile valori.

Ultimul useEffect contine filtrarea anunturilor si se va apela de fiecare data cand se schimba filtrele, adica variabila filters, respectiv setul de anunturi, variabila excelData.

Aceasta componenta va returna conditionat in functie de numarul de anunturi, 3 campuri cu etichete sugestive destinate valorilor filtrelor si un tabel cu anunturile care va avea toate coloanele care se regasesc si in excelul de pe disc, ultima coloana din acest tabel va fi reprezentata de un buton care ne va deschide o noua pagina catre link-ul anuntului de unde a fost extras pentru mai multe detalii despre acesta.

### Componenta administratorului

Ca si administrator putem face urmatoarele lucruri: sa actualizam setul de date destinat anunturilor imobiliare din Cluj cat si din Bucuresti si sa reantrenam modelul de predictie a pretului manual. Functionalitatile mentionate anterior se regasesc implementate in componenta AdminPage.

Componenta AdminPage este o functie care returneaza interfata vizuala a paginii de administrare sub format JSX, pentru asigurarea functionalitatilor dorite am avut nevoie de trei variabile de stare declarate folosin useState:

* isLoading – este o variabila de tip bool si reprezinta starea de incarcare care va fi false cat timp nu se colecteaza noi anunturi, respective true in timpul colectarii, aceasta variabila ne ajuta sa reprezentam vizual perioada in care se colecteaza anunturile.
* nrAnunturiColectate – este o variabila de tip int si reprezinta numarul de inregistrari colectate in urma actiunii de actualizare a setului de date
* city – este o variabila de tip string , initializata cu valoarea cluj, care ne ajuta sa identificam pentru ce oras se doreste actualizarea setului de date.

Functia handleExtraction este apela atunci cand utilizatorul apasa pe butonul Actualalizeaza pentru a colecta anunturi din orasul selectat. In interiorul functie se va construi URL-ul pentru cerere in functie de valoarea variabile city. Daca setul de date a fost actualizat cu succes atunci se va afisa numarul de anunturi colectate primit ca raspuns de la server, iar daca nu , se va afisa un mesaj de eroare in cosola.

Functia onCityChange este apelata de fiecare data cand orasul selectat se schimba si in interiorul ei se realizeaza setarea variabile de stare city la valoarea selectata in interfata.

Aceasta componenta returneaza un div in care se gaseste un selectBox pentru selectarea orasului cu optiunile Bucuresti sau Cluj si un buton pentru actualizarea setului de date. Cat timp are loc colectarea datelor se va afisa textul :Loading... , iar la finalul colectarii se va afisa cate anunturi au fost colectate.

La final aceasta componenta este exportata pentru a putea fi folosita si in alte parti , mai exact la crearea rutelor aplicatiei. ????

### Comoponenta de statistici

Componenta Statistici returneaza un meniu cu toate reprezentarile grafice generate, si in functie de optiuniea pe care o vom selecta ni se va afisa o poza cu graficul respectiv si o scurta descriere.

Mai intai am importat toate imaginile cu grafice din folderul assets, dupa care am declarat o constanta globala de tip obiect care are cate o eticheta sugestiva pentru fiecare grafic disponibil, ca si valoare pentru aceste etichete avem tot un obiect cu atributele: image care reprezinta imaginea propriu-zisa cu graficul si description, descrierea care va fi afisata impreuna cu acesta.

Ca si variabile de stare interne aceasta componenta are doar starea selectedOption folosita pentru a identifica optiunea selectata din meniu. Functia handleOptionSelected se apeleaza de fiecare data cand se selecteaza o alta optiune si are rolul de a seta variabila seletedOption.

Functia renderGraph se ocupa cu randarea oricarui grafic in interfata. Aceasta verifica daca avem vreo optiune selectata, daca nu avem se va afisa un mesaj sugestiv, iar daca avem vom extrage imaginea si descriere in doua constante locale din obiectul global care le contine si vom returna un div compus din o imagine si un paragraf cu descrierere.

### Componenta pentru estimarea pretului

Componenta PriceEstimate consta intr-un formular in care utilizatorul introduce caracteristicile proprietatii sale, iar apoi folosindu-ne de modelele de predictie create ii vom prezice un pret.

In partea de sus a componentei functionale se declara variabilele de stare necesare estimării pretului, acestea sunt : (price, priceRF, priceSVM) cate o variabila pentru fiecare pret prezis de fiecare model diferit, variabila care stocheaza cartierele din JSON (neighborhoods), și pentru datele introduse de utilizator (formData).

Functia handleChange este o funcție pentru a actualiza starea formData atunci când utilizatorul completează câmpurile de intrare. Ea este apelată atunci când valorile din câmpurile de intrare sunt modificate.

Functia handleSubmit este apelată atunci când utilizatorul trimite formularul. Ea face o cerere POST către serverul backend (la adresa http://localhost:5000) cu datele introduse de utilizator în formData.Când serverul răspunde cu rezultatele estimărilor de preț, valorile sunt actualizate în starea componentei.

Efectul useEffect este folosit pentru a încărca datele din mapare\_locatii.json în starea neighborhoods la încărcarea inițială a paginii.

Componenta este compusa dintr-un formular care contine pentru fiecare caracteristică (număr de camere, suprafață utilă etc.) câmpuri de intrare pentru ca utilizatorii să introducă valorile.

Câmpul "Cartier" este reprezentat de un meniu select, unde utilizatorii pot alege unul din cartierele încărcate din neighborhoods.

Pentru ca utilizatorul sa trimita formularul se foloseste un buton Estimare Pret, cand acesta este apasat se afișează rezultatele estimărilor de preț pentru fiecare model, precum și o medie și preț total estimat.

### Componenta principală

La inceputul componentei principale sunt importate fișierele necesare pentru funcționarea componentei App. Menu este o componentă pentru meniu, Route și Routes sunt componente oferite de React Router pentru gestionarea rutelor și navigarea între pagini, iar celelalte importuri sunt pentru diverse funcționalități și hook-uri.

Apoi este creat un context numit AppContext. Acest context va fi folosit pentru a partaja informații între diferite componente din aplicație.

Componenta App reprezinta nucleul aplicatiei Aici sunt definite stările isLoggedIn (pentru verificarea autentificării utilizatorului) și isAdmin (pentru a verifica dacă utilizatorul este administrator). De asemenea, se inițializează variabilele de stare userId și setUserId utilizând hook-ul usePersistentState, care pare să fie definit în altă parte a codului.

Componenta utilizează contextul AppContext.Provider pentru a furniza aceste informații altor componente din aplicație. În interiorul acestei componente, se randează meniul (Menu) și rutele aplicației (appRoutes) folosind componente Routes și Route. Acestea din urmă sunt gestionate de către React Router pentru a direcționa utilizatorii către paginile corespunzătoare.

### Alte componente importante

In cadrul pachetului constants am creat fisierul **PagesUrl** care contine toate rutele disponibile in aplicatie, deoarece acestea pot aparea in mai multe componente ale aplicatiei declarandu-le intr-un singur loc centralizat vom asigura coerenta si usurinta in gestionarea acestora . Rutele sunt definite in obiectul PAGES\_URL sub forma de perechi cheie-valoare , cheia reprezinta un identificator unic pentru o anumita ruta, in timp ce valoarea reprezinta ruta efectiva care se va afisa in browser. De exemplu, identificatorul Admin este folosit pentru ruta catre pagina administratorului si este setata la /admin.

Componenta **usePersistentState** din pachetul hooks reprezinta un hook personalizat care are ca si scop pastrarea unei stari intre sesiuni si reincarcari ale aplicatiei, mai exact a fost implementata pentru pastrarea state-urilor legate de statusul de login al utilizatorilor, astfel oferindu-le acestora o experienta coerenta fara delogari involuntare.

Aceasta functie are doi parametrii, storageKey care reprezinta cheia la care se gaseste valoarea starii dorite in localStorage, respectiv initialState care reprezinta valoarea initiala a variabilei de stare.

Mai intai vom declara o variabila de stare interna persistentState folosind useState() si ii vom oferii ca si valoare initiala parametrul primit de functie.

Iar apoi prin intermediul unui useEffect care se va executa o singura data la incarcarea initiala a componentei vom verifica valorile stocate in localStorage la cheia storageKey. Daca exista o valoarea salvata in localStorage, aceasta este preluata si se verifica daca are valoarea true sau false , daca da, atunci se va transforma intr-un boolean, altfel, se pastreaza ca si sir de caractere. Daca exista o valoare recuperata din localStorage atunci starea interna va fi setata cu acea valoare.

In ultima parte a codului se defineste o functie setState care va inlocui functia obisnuita de setare a straii , aceasta va actualiza starea interna persistentState dar va si salva valoarea acesteia in localStorage la cheia respectiva.

Acest hook va returna un array format din doua elemente: persistentState- starea curenta persistenta si setState- functia pentru actualizarea starii.

???? ??validatorul de email

## Aplicatia de gestiune a utilizatoriilor

### Descrierea claselor si metodelor

#### Clasa User

Clasa User- reprezinta entitatea de baza si incapsuleaza diverse atribute care definesc un utilizator. Aceste atribute sunt : un identificator unic de tip UUID(id), prenumele (firstname), numele de familie (lastname), adresa de e-mail (email), parola (pasword) si rolul (role) de tip string. Dupa declarearea atributelor, clasa User prezintă un constructor care inițializează obiectul cu valorile furnizate, iar ca si metode avem doar getter si setter pentru fiecare atribut.

#### Interfata UserRepository

Interfata UserRepository extinde JpaRepository oferit de frameworkul Spring pentru a facilita interactiunea dintre aplicatie si baza de date. Aceasta mosteneste operatiile predefinite pentru inregistrarile de tip user, cum ar fi: salvarea, preluarea, actualizarea și ștergerea datelor utilizatorului.

#### Clasa UserBuilder

Clasa UserBuilder nu are atribute servind ca o clasa de utilitate. Clasa are două metode esențiale: toUserDTO() și toEntity(). Metoda toUserDTO() convertește un obiect User într-un obiect UserDTO corespunzător, facilitând transferul de date și straturile de abstractizare în cadrul aplicației. În schimb, metoda toEntity() permite conversia unui obiect UserDTO înapoi într-un obiect User.

#### Clasa UserDTO

Clasa UserDTO extinde RepresentatiomModel si contine toate atributele clasei user, dar si metodele de get si set, in plus avem un constructor fara parametrul id, aceasta clasa suprascrie metodele hashCode si equals, metoda equals verifica daca un obect de tip object este egal cu un obiect de tip user, iar metoda hash returneaza codul hash al obiectului.

#### Clasa LoginDTO

Clasa LoginDTO este folosita pentru autentificarea si include doar atributele necesare procesului de conectare, si anume password si email. In aceasta clasa se regaseste un constructor si metodele getter si setter pentru fiecare atribut.

#### Clasa UserController

Clasa UserController are un atribut final de tip UserService care faciliteaza interactiunea intre controller si serviciile corecpunzatoaare. Clasa contine doua metode , cea de login si register, in interiorul metodelor se apeleaza metodele din service corespunzatoare pentru autentificare, respectiv inregistrare si se retuneaza un ResponseEntity adecvat. Ambele metode au adnotari de @PostMapping cu numele rutei permitand aplicatiei sa gestioneze cererile POST primite si sa execute logica asociata.

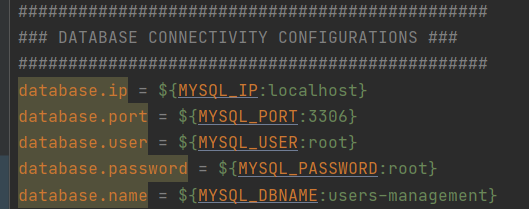
#### Clasa UserService

Clasa UserService se foloseste de un obiect de tip UserRepository pentru a implementa logica din spatele metodelor de login si register.

### Persistenta datelor

În cadrul aplicației, se utilizează baza de date MySQL pentru a asigura persistența datelor. Acest aspect este deosebit de important, deoarece permite stocarea și accesarea informațiilor relevante pentru utilizatori într-un mod sigur și fiabil.

Prin intermediul framework-ului Spring și a bibliotecii Hibernate, aplicația este configurată pentru a comunica cu baza de date MySQL. Acest lucru implică definirea unei conexiuni la baza de date și a unui set de proprietăți, precum utilizatorul și parola necesare pentru autentificarea la serverul MySQL. Aceste proprietati se regasesc in fisierul application.properties si trebuie schimbate in functie de credentialele proprii.



Datorita faptului ca aplicatia se ocupa doar cu gestiunea utilizatorilor, in baza de date vom avea un singur tabel, acesta este tabelul users care are urmatoarele coloane :id, firstname, lastname, email, password si role, coloana id reprezentand cheia primara.

# Testare și validare

## Testarea si validarea modelelor de predictie

## Testarea si validarea modulului care colecteaza anunturi

Pentru testarea functionalitatii de colectare de anunturi de pe Internet vom merge pe pagina administratorului si vom apasa butonul ‘actualizeaza’ pentru a incepe colectarea. Apoi, daca procesul s-a terminat cu succes, vom vizualiza inregistrarile din fisierul output.csv. Pentru validarea functionarii corecte a acestui modul analizam valorile stocate in fiecare coloana, ne asiguram daca o anumita coloana are un tip de valoare valid , de exemplu daca pe coloana ‘Nr. Camere’ este scris un numar si nu valoarile altei caracteristici cum ar fi cartierul din care este anuntul. De asemenea, chiar daca valorile sunt de tipul corect, pentru validare am luat in considerare si sa nu fie prezente valori aberante, deoarece dupa prima testare am observat ca pentru coloana pretului aveam o valoare foarte mica, ceea ce ridica anumite semne de intrebare, in final realizand faptul ca un punct pus pe website-ul respectiv pentru o vizualizare mai buna a pretului era transpus in virgula la salvarea datelor. Ultimul pas de validare consta in navigarea catre link-ul salvat pe coloana ‘Link anunt’ pentru a verifica daca valorile caracteristicilor sunt exact cele salvate in excel.

Mai jos se poate observa on inregistrare din tabela de anunturi si datele din website care coincid.

Acest capitol, împreună cu cele două care îl preced, va reprezenta aproximativ 70% din lucrare.

# Manual de instalare si utilizare

În cadrul capitolului curent se vor detalia resursele necesare (atât software, cât și hardware) pentru instalarea și rularea sistemului, dar se va include și o descriere pas cu pas a procesului de instalare.

Totodată, în cadrul acestui capitol, se va descrie modul de utilizare al aplicației din punct de vedere al utilizatorului, fără a se menționa aspecte tehnice interne, ci doar

cunoștințe de bază deținute de orice utilizator neexperimentat.

## Resurse necesare

Pentru a putea utiliza aceasta aplicatie vom avea nevoie de urmatoarele resurse software:

Resursa software de bază este ca pe calculator să fie prezent un sistem de operare Windows 7 sau 10 m pentru a putea instala restul resurselor software necesare

In plus, vom avea nevoie să instalăm următoarele aplicații: IntelliJ, Visual studio code, Python, Node.js, MySQL Workbench. Instalarea acestora se va face dupa manualul de instalare dedicat de pe site-ul fiecarei aplicatii in parte.

Pentru instalarea dependențelor interne putem utiliza diverse comenzi rulate în terminal cum ar fi npm install pentru aplicația web, pip install pentru aplicația python iar cele din aplicația inteligenței se vor instala automat.

Această aplicație va putea fi transferată de pe un calculator pe altul prin intermediul unui stick sau prin intermediul diferitelor platforme precum gitHub de unde se va putea descărca codul.

După dezarhivarea codului vom regăsi 3 foldere, cel cu numele userManagement va conține soluția scrisă în IntelliJ, folderul cu numele prediction a conține aplicația web și predicție va conține aplicația parton care va fi deschisă folosind deja studio cod.

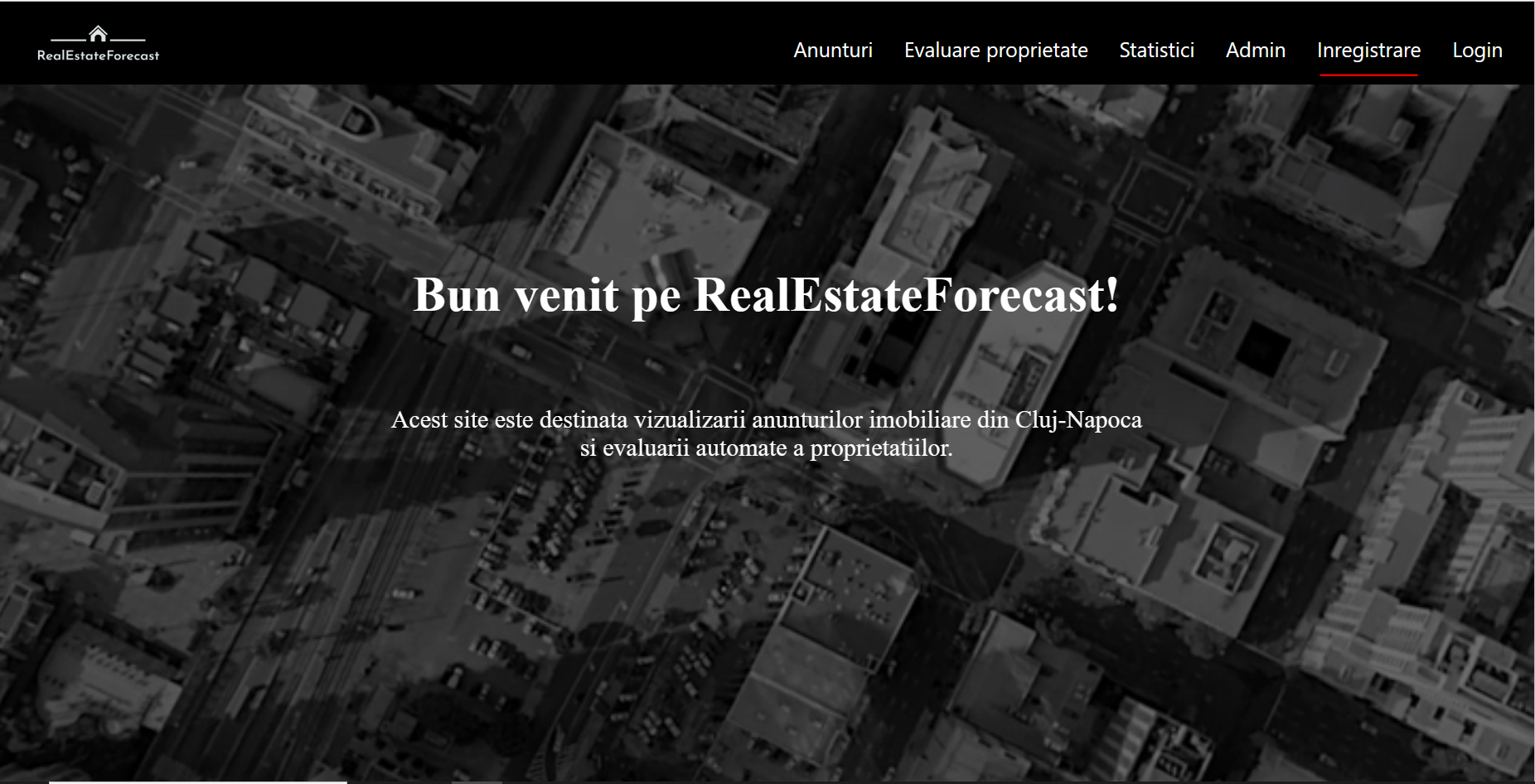
Pentru a rula întreaga soluție vom rula fiecare aplicație în parte.

Primul pas este acela de a deschide aplicația MySQL Workbranch si de a crea o noua baza de date cu numele users-management

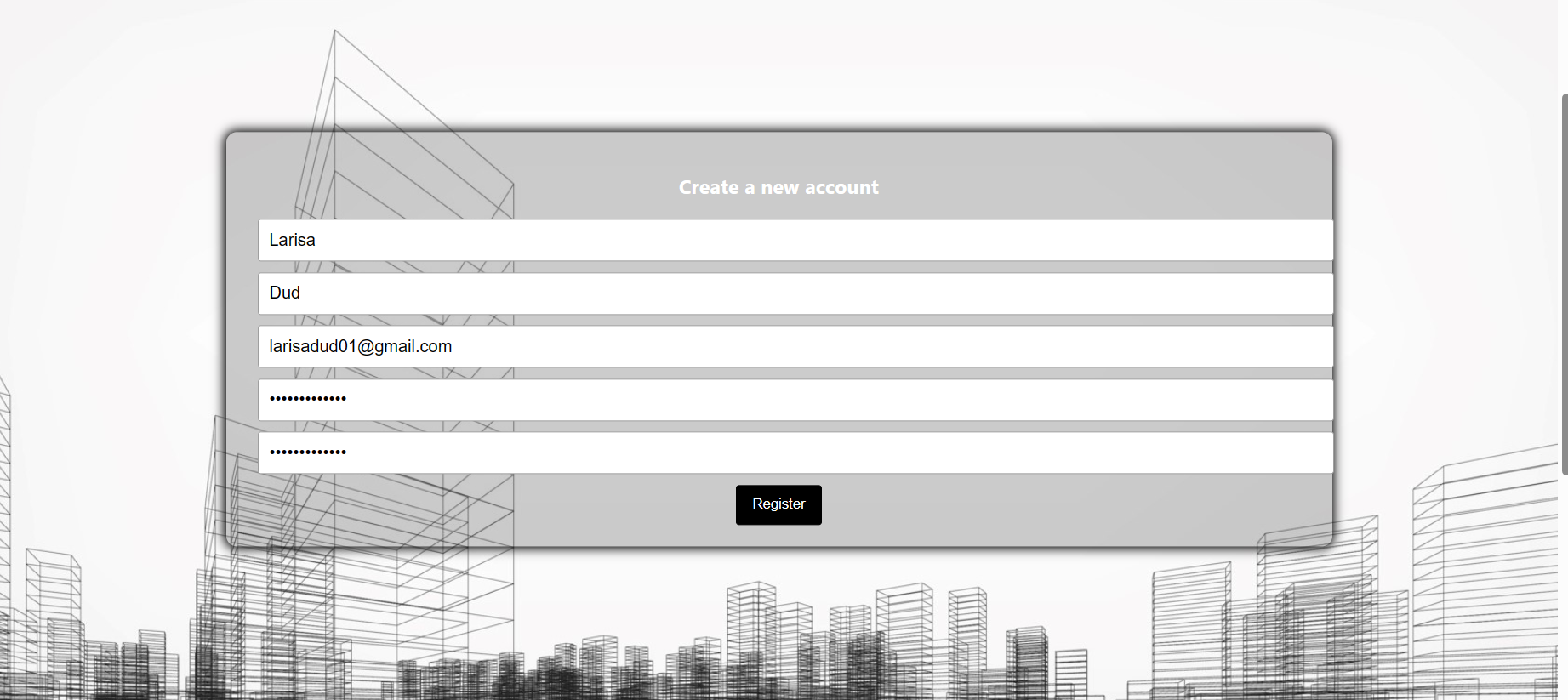
Pentru a rula aplicația de gestiune a utilizatorilor vom deschide mediul de programare IntelliJ, vom apăsa pe opțiunea fail apoi pe open iar în caseta deschisă vom selecta ruta către folderul PricePrediction și vom apăsa open. După ce proiectul este deschis și s-au instalat toate dependințele vom apăsa pe butonul din figura de mai jos.

## Manual de utilizare

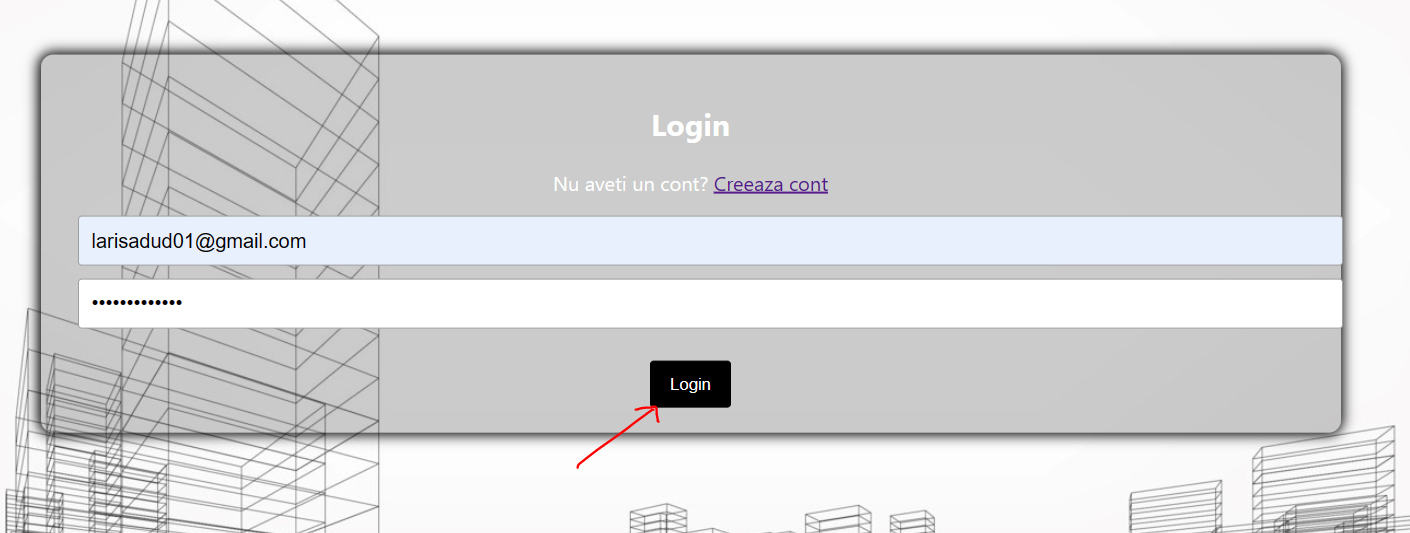
Pentru a ne înregistra un cont în aplicație trebuie să apăsăm pe opțiunea înregistrare din meniu.



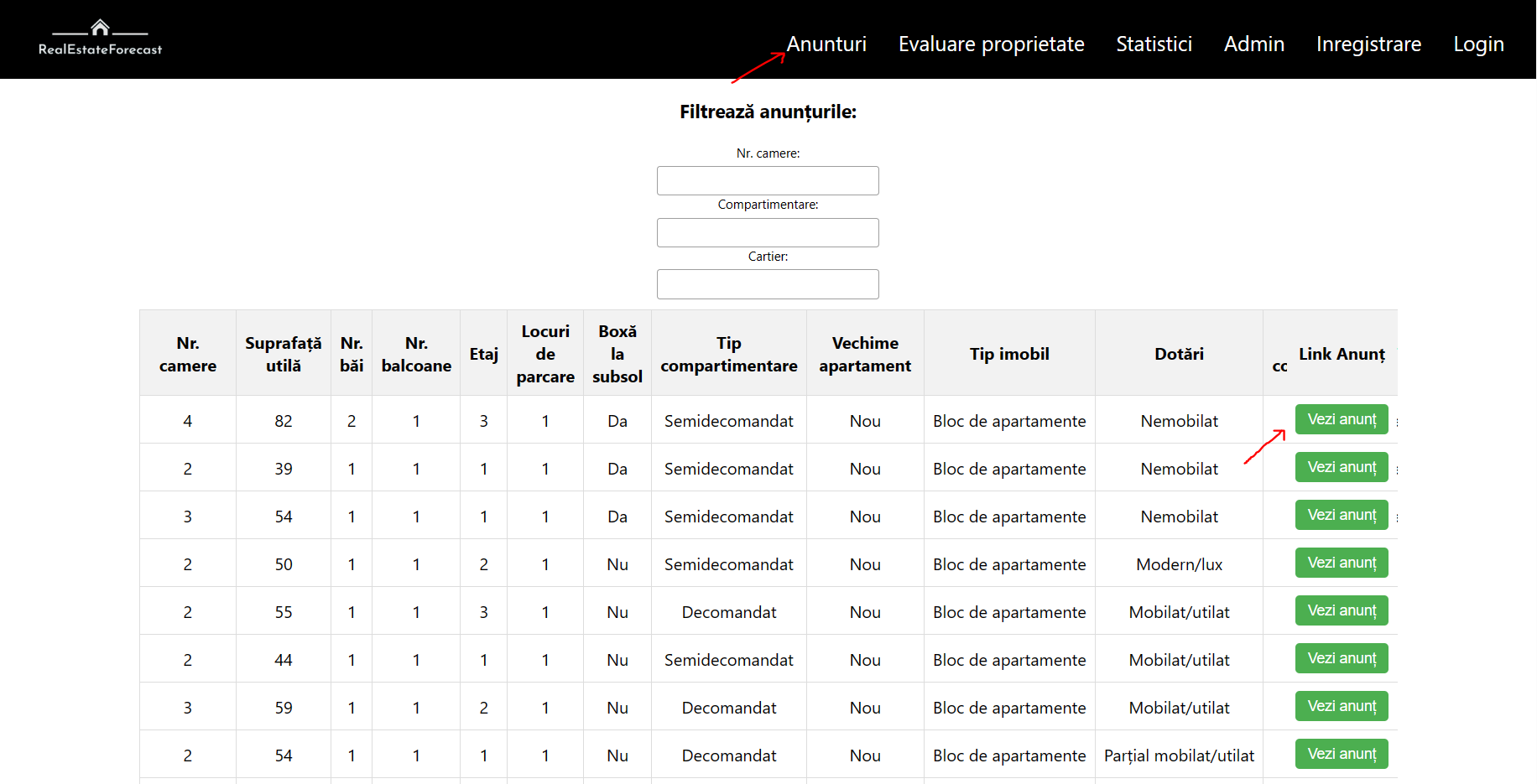
Apoi vom fii redirectati către pagina cu formularul de înregistrare unde trebuie să completăm toate câmpurile cu date valide și să apăsăm butonul register.



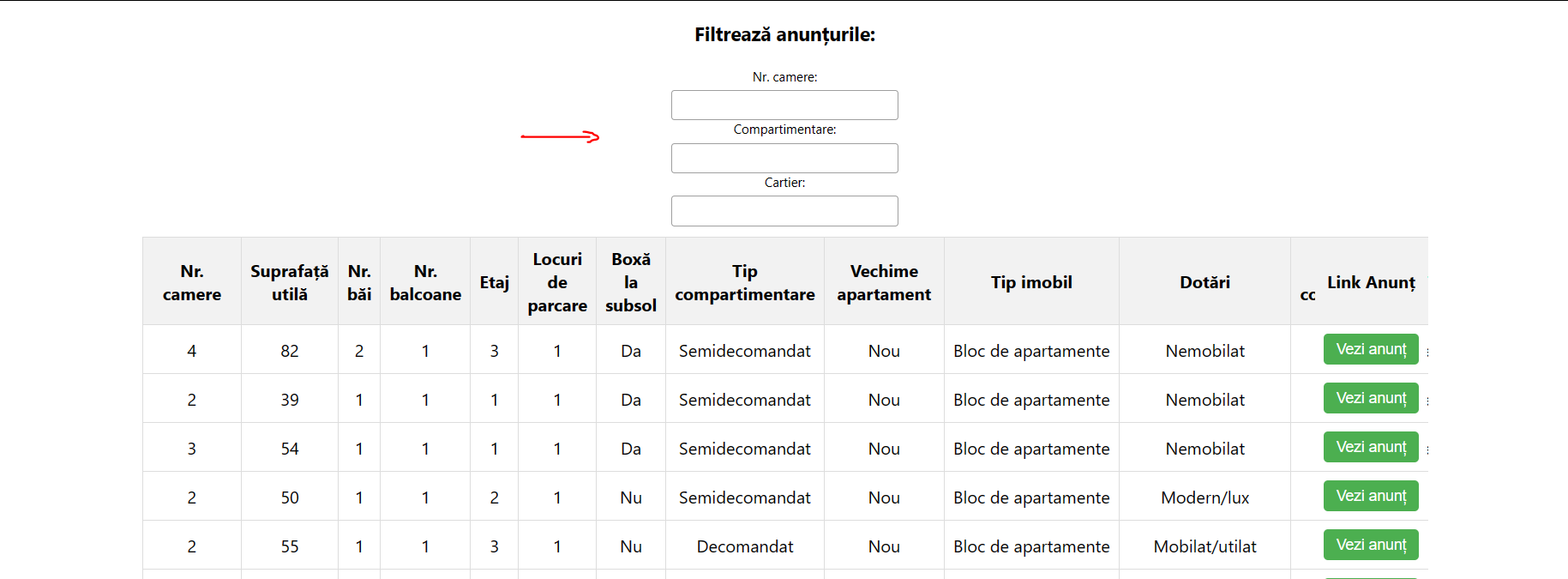
Pentru a ne autentifica în aplicație trebuie să apăsăm pe opțiunea login din meniu care ne va redirecta către pagina de login unde trebuie să introducem credențialele noastre și să apăsăm butonul de login.



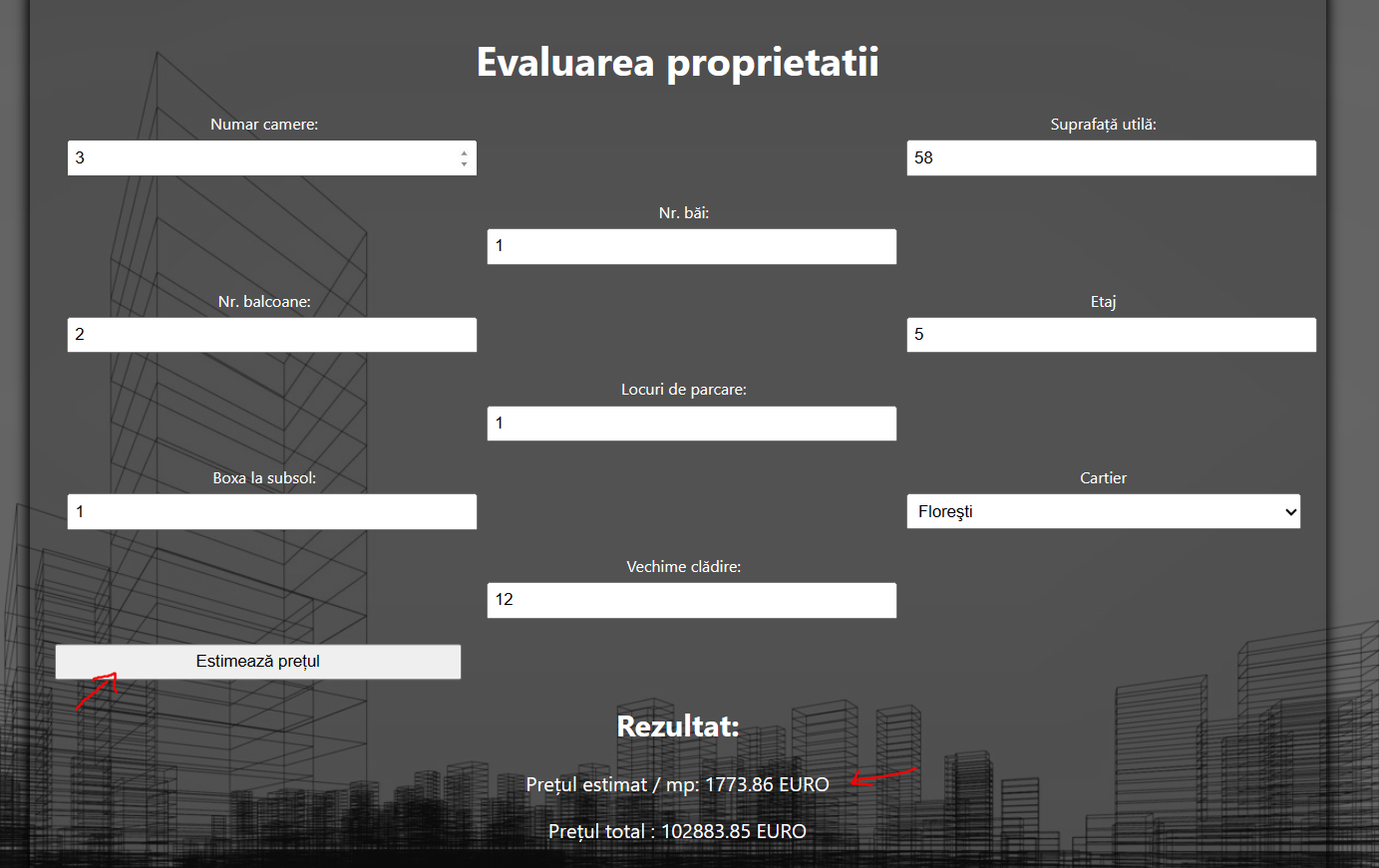
Pentru a vizualiza anunțurile disponibile în setul de date vom selecta opțiunea anunțuri din meniu fiind redirectați către pagina cu anunțuri iar pentru a vedea mai multe detalii despre un anumit anunț din tabel vom da click pe butonul vezi anunț din dreptul fiecărei înregistrări și se va deschide o nouă pagină cu anunțul original.



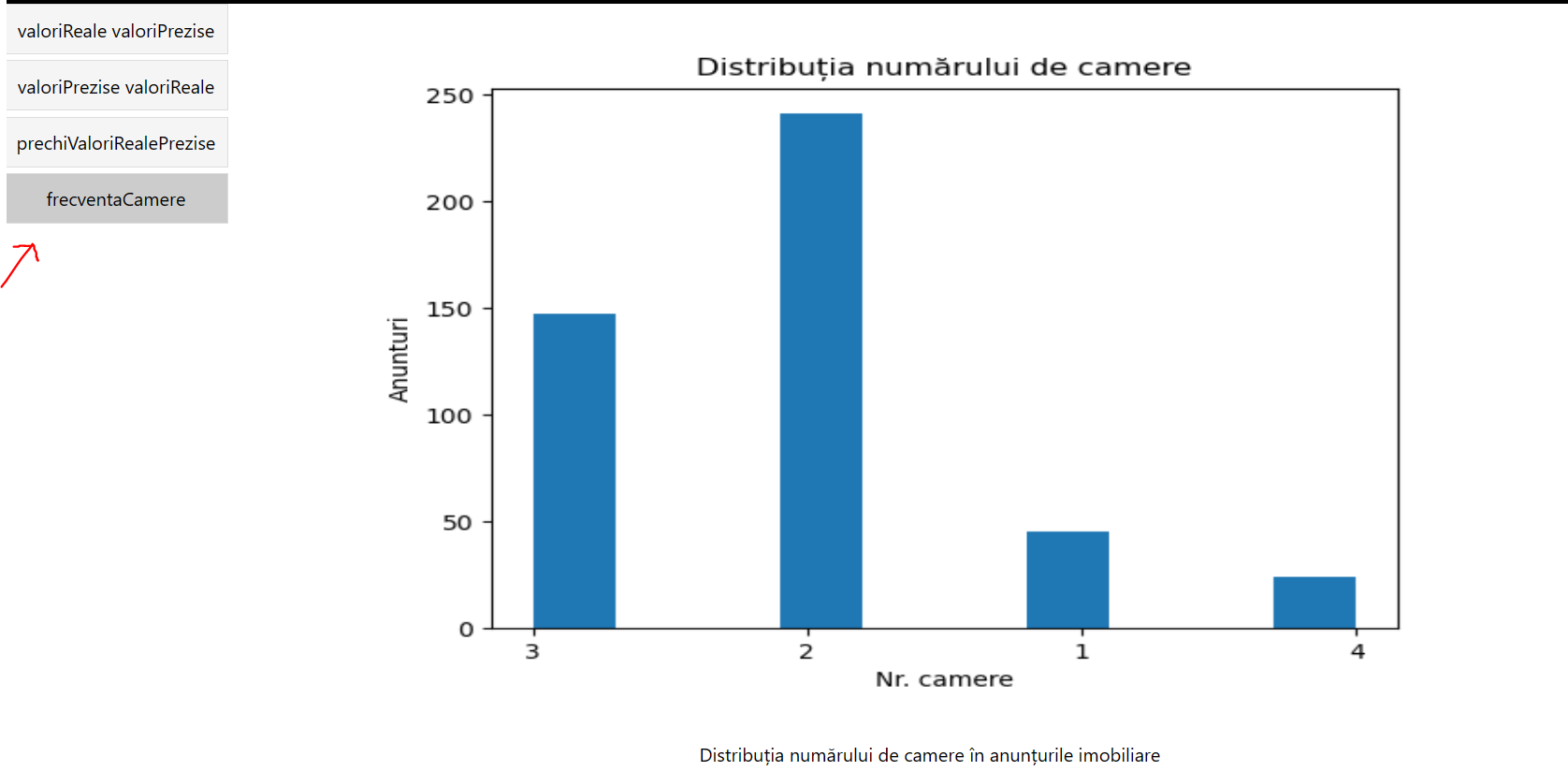
Tot pe pagina de anunțuri pentru a filtra anunțurile afișate trebuie să introducem caracteristicile dorite în câmpurile din secțiunea filtrează anunțurile iar acestea se vor filtra automat fără să fie nevoie de alte acțiuni.



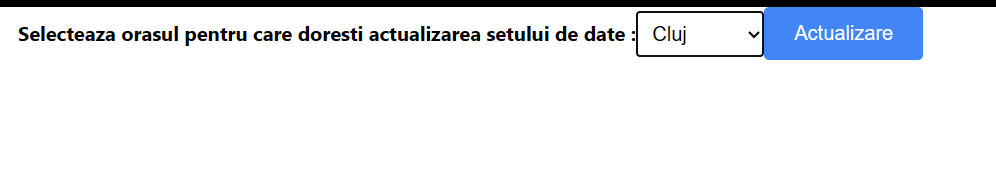
Dacă utilizatorul dorește să evalueze o proprietate va da click pe opțiunea evaluare proprietate din meniu acesta fiind redirectat către pagina corespunzătoare acestei acțiuni. Pe pagina de evaluare, utilizatorul va introduce valorile caracteristicilor proprietății sale în câmpurile aferente iar apoi va apăsa butonul estimează preț urmând ca prețul să fie afișat în secțiunea rezultat.



Pentru vizualizarea anumitor statistici utilizatorul trebuie să dea click pe opțiunea statistici din meniu acesta fiind redirectat către o pagină care conține un meniu cu toate graficele disponibile iar pentru a le vizualiza trebuie să facă clic pe opțiunea dorita.



Pentru a actualiza setul de date folosit pentru modelul de predicție administratorul va accesa pagina destinat strict lui făcând click pe opțiunea admin din meniu după care dintr-un select box selectează dacă colectarea se va face pentru setul de date din Cluj sau din București și apasă butonul actualizare.



Pentru a ne deloga trebuie să facem click pe opțiunea logout din meniu.

Minimum 1 pagină, până la 5 pagini

# Concluzii

Acest capitol va include concluziile ce se pot trage in urma realizarii acestei lucrari prezentand o imagine de ansamblu asupra contributiilor noastre, o analiza critica a rezultatelor obtinute si posibile cai de dezvoltare si imbunatatiri ulterioare.

## Contribuții personale

## Analiza critică a rezultatelor obținute

## Posibile dezvoltări și îmbunătățiri ulterioare

Acest capitol va ocupa 1-2 pagini.

Capitolul ar trebui sa conțină (nu se rezumă neapărat la):

* un rezumat al contribuțiilor voastre
* o analiză critică a rezultatelor obținute
* o descriere a posibilelor dezvoltări și îmbunătățiri ulterioare

# Bibliography

|  |  |
| --- | --- |
| [1] | G. Boella și L. van der Torre, „Contracts as Legal Institutions in Organizations of Autonomous Agents,” în *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, New, 2004. |
| [2] | G. Boella, J. Hulstijn și L. van der Torre, „A Synthesis Between Mental Attitudes and Social Commitments in Agent Communication Languages,” în *Intelligent Agent Technology 05 (IAT 2005)*, Compiegne, 2005. |
| [3] | G. Cachon și M. Lariviere, „Supply chain coordination with revenue sharing contracts: strengths and limitations,” *Management Science,* vol. 51, pp. 30-44, 2005. |
| [4] | C. P. Pfleeger, S. L. Pfleeger și J. Margulies, Security in Computing, 5th Edition, Pearson, 2015. |
| [5] | Software Freedom Conservancy, „The Selenium Browser Automation Project,” [Interactiv]. Available: https://www.selenium.dev/. [Accesat 17 martie 2021]. |

**[6] https://www.kaggle.com/datasets/denisadutca/bucharest-house-price-dataset**

[**https://www.ibm.com/topics/linear-regression**](https://www.ibm.com/topics/linear-regression)

<https://www.sciencedirect.com/topics/medicine-and-dentistry/linear-regression-analysis>

<https://books.google.ro/books?hl=ro&lr=&id=IvAw_1MTASsC&oi=fnd&pg=PA3&dq=linear+regression&ots=GtmF6ofvnC&sig=M5th0al3eySI7Z4xpYfEX59U9uo&redir_esc=y#v=onepage&q=linear%20regression&f=false>

După ce ați inserat/actualizat bibliografia selectați întregul tabel și aplicați stilul *Biblio.*  Stilul *Normal* are indentare la începutul paragrafelor și, de aceea nu veți obține formatul ca mai sus fără acest pas.

# Anexa 1

…

Secțiuni de cod relevante

# Anexa 2

Alte informații relevante (demonstrații etc.)

…

# Anexa 3

Lucrări publicate (dacă există)

etc.