



UNIVERSITATEA DIN  
BUCUREŞTI



FACULTATEA DE  
MATEMATICĂ ŞI  
INFORMATICA

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

**INTERIO - APLICAȚIE MOBILĂ DE  
DESIGN INTERIOR  
PERSONALIZAT PE BAZA  
INTELIGENȚEI ARTIFICIALE ȘI A  
TESTELOR DE PERSONALITATE**

Absolvent  
Potlog Ioana

Coordonator științific  
Prof. Dr. Istrate Florentin

București, iunie-iulie 2025

## Rezumat

*Interio* este o aplicație mobilă care le permite utilizatorilor să genereze imagini realiste cu propriile camere redecorate, în stiluri și palete de culori personalizate. Procesul începe cu completarea unui chestionar care explorează aspecte legate de stilul de viață, preferințele estetice și trăsăturile de caracter ale utilizatorului. Pe baza răspunsurilor oferite, un clasificator AI identifică trei stiluri de design interior care se potrivesc cel mai bine profilului acestuia, alături de paletete de culori corespunzătoare. În continuare, utilizatorul poate alege dintre două modalități de configurare a camerei: încărcarea unei fotografii reale a unei încăperi existente sau construirea unei schițe, cu ajutorul unui editor vizual interactiv ce permite plasarea manuală a obiectelor de mobilier. În ambele cazuri, imaginea camerei este transmisă unui model AI care generează automat o versiune redecorată, adaptată stilului și preferințelor selectate. Această funcționalitate oferă o experiență interactivă, ajutând utilizatorii să vizualizeze modul în care spațiul din locuința lor ar putea arăta, într-un stil care îi reprezintă.

## Abstract

*Interio* is a mobile application that allows users to generate realistic images of their own rooms, redecorated in personalized styles and color palettes. The process begins with a quiz that explores aspects such as lifestyle, aesthetic preferences, and personality traits. Based on the provided answers, an AI classifier identifies the three interior design styles that best match the user's profile, along with the corresponding color palettes. Next, the user can choose between two methods for configuring the room: either by uploading a real photo of an existing space or by building a schematic layout using an interactive visual editor that allows manual placement of furniture items. In both cases, the resulting image is sent to an AI model that automatically generates a redecorated version, tailored to the selected style and preferences. This functionality provides an interactive experience, helping users visualize how their living space could look in a style that reflects who they are.

# Cuprins

<b>1 Introducere</b>	<b>5</b>
1.1 Tipul lucrării și subdomeniul specific . . . . .	5
1.2 Prezentarea generală a temei . . . . .	5
1.3 Scop și motivație . . . . .	5
1.4 Contribuția proprie . . . . .	6
1.5 Structura lucrării . . . . .	6
<b>2 Preliminarii</b>	<b>7</b>
2.1 Noțiuni teoretice fundamentale . . . . .	7
2.1.1 Arhitectura client-server . . . . .	7
2.1.2 Structurarea aplicației pe straturi ( <i>n-tier architecture</i> ) . . . . .	7
2.1.3 Persistența datelor cu ORM . . . . .	7
2.1.4 Model de clasificare - Random Forest . . . . .	8
2.1.5 Modele de difuzie pentru generarea imaginilor . . . . .	9
2.1.6 Control Net . . . . .	9
2.2 Tehnologii utilizate . . . . .	9
2.2.1 Frontend - React Native . . . . .	9
2.2.2 Backend - ASP.NET Core . . . . .	10
2.2.3 Baza de date - MSSQL . . . . .	11
2.2.4 Tehnologii AI . . . . .	11
2.2.5 Alte tehnologii și instrumente auxiliare . . . . .	11
2.3 Contextul actual . . . . .	12
2.4 Obiective generale ale lucrării . . . . .	13
<b>3 Contribuția proprie</b>	<b>14</b>
3.1 Arhitectura aplicației . . . . .	14
3.2 UI și UX . . . . .	15
3.3 Autentificare . . . . .	16
3.3.1 Pagina de Sign up și Login . . . . .	16
3.3.2 Resetarea parolei . . . . .	16
3.4 Chestionarul de personalitate și preferințe . . . . .	17
3.4.1 Scopul și implementarea chestionarului . . . . .	17
3.4.2 Crearea dataset-ului de antrenare . . . . .	18

3.4.3	Implementarea clasificatorului de stiluri . . . . .	18
3.5	Pagina principală și de profil . . . . .	19
3.6	Decorarea unei camere . . . . .	20
<b>4</b>	<b>Concluzii</b>	<b>24</b>
	<b>Bibliografie</b>	<b>25</b>
	<b>Anexe</b>	<b>27</b>
	Anexa 1 - Interfața aplicației . . . . .	27
	Anexa 2 - Secvențe de cod . . . . .	30
	Anexa 3 - Structura bazei de date . . . . .	31

# Listă de figuri

Figura 3.1. -	Arhitectura aplicației . . . . .	14
Figura 3.2. -	Evaluarea clasificatorului Random Forest . . . . .	19
Figura 3.3. -	Interfața pentru alegerea imaginii de start (fotografie reală sau schiță) . . . . .	21
Figura 3.4. -	Pașii următori în generarea imaginii cu camera redecorată . . . . .	21
Figura 3.5. -	Comparație între metodele de condiționare a imaginii initiale în ControlNet . . . . .	23
Figura 1. -	Fluxul UI al funcționalității de autentificare . . . . .	27
Figura 2. -	Fluxul UI al funcționalității de resetare a parolei . . . . .	28
Figura 3. -	Fluxul UI al funcționalității de gestionarea a profilului utilizatorului . . . . .	28
Figura 4. -	Fluxul UI al funcționalității de generare a imaginii cu camera redecorată . . . . .	29
Figura 5. -	Secvență de cod — clasificator Random Forest . . . . .	30
Figura 6. -	Secvență de cod — generarea imaginilor cu Stable Diffusion și ControlNet . . . . .	30
Figura 7. -	Diagrama entitate-relație (ERD) — structura bazei de date . . . . .	31

# 1. Introducere

## 1.1 Tipul lucrării și subdomeniul specific

Lucrarea se încadrează în subdomeniul aplicațiilor software și constă în realizarea unei aplicații mobile menite să rezolve o problemă clar definită.

## 1.2 Prezentarea generală a temei

*Interio* este o aplicație mobilă care îmbină tehnologia inteligenței artificiale cu designul interior, oferindu-le utilizatorilor o experiență personalizată pentru amenajarea locuinței. Principalul avantaj al aplicației constă în capacitatea de a genera imagini realiste privind decorarea unei camere goale sau redescorarea uneia deja mobilată.

Spre deosebire de alte soluții existente, *Interio* oferă sugestii de stiluri de decor adaptate profilului fiecărui utilizator, inspirate din trăsăturile de personalitate identificate prin intermediul unui chestionar interactiv. Aceste explorează aspecte precum stilul de viață, valorile personale și preferințele estetice, contribuind la conturarea unui profil. Pe baza răspunsurilor, algoritmul AI clasifică și returnează un set de trei stiluri de design interior pe baza preferințelor utilizatorului, împreună cu o paletă de culori asociată fiecărui stil. În plus, aplicația include și o pagină informativă dedicată stilurilor recomandate, oferind detalii vizuale și descriptive care îl ajută pe utilizator să înțeleagă mai bine caracteristicile fiecărei opțiuni propuse.

## 1.3 Scop și motivație

Scopul aplicației *Interio* este de a le oferi utilizatorilor o experiență personalizată și intuitivă de amenajare a uneia sau mai multor camere. Prinț-un proces simplu utilizatorii pot obține recomandări stilistice adaptate profilului personal și pot vizualiza rezultatul final direct în aplicație, sub forma unor imagini realiste generate pe baza fotografiilor încărcate din galeria telefonului mobil sau a schițelor create folosind instrumentul de editare și aranjare a mobilei. *Interio* își propune să eliminate incertitudinea și timpul pierdut cu alegerea unui stil de design, transformând procesul de decorare într-o activitate intuitivă și interactivă, bazată pe personalitatea utilizatorului.

Motivația din spatele dezvoltării aplicației a pornit din dorința de a îmbina două arii de interes personal: dezvoltarea de aplicații mobile și domeniul creativ al designului interior. Observând interesul tot mai mare pentru aplicațiile care oferă recomandări personalizate, am remarcat că, în domeniul ales, opțiunile existente sunt încă destul de limitate. Așa a apărut ideea de a crea o soluție care să sprijine utilizatorii în procesul de amenajare,

nu doar prin generarea de imagini, ci și prin oferirea unor sugestii adaptate personalității fiecărui, cu ajutorul unui chestionar și a unui algoritm de clasificare.

## 1.4 Contribuția proprie

Aplicația *Interio* se remarcă printr-o serie de componente tehnice și concepte originale, dezvoltate pentru a oferi o experiență particularizată în amenajarea locuințelor.

Ideea principală a aplicației constă în capacitatea utilizatorului de a genera o imagine realistă a unei camere redeterminate, fie prin încărcarea unei fotografii reale a propriei camere, fie prin construirea unei schițe pornind de la o cameră goală, folosind un editor vizual interactiv. În această variantă, utilizatorul poate adăuga și poziționa manual obiecte de mobilier dintr-o colecție predefinită de elemente. Aceasta selectează manual tipul camerei, stilul dorit și paleta de culori dintre cele recomandate, iar apoi toate aceste opțiuni sunt utilizate pentru construirea unui prompt complex, care este trimis către modelul de inteligență artificială responsabil de generarea unei versiuni redeterminate a camerei.

Contribuția mea originală constă în integrarea unui sistem intelligent de potrivire între utilizator și stilul de decor interior. Am creat un chestionar care explorează preferințele estetice și trăsăturile de personalitate ale utilizatorului. Clasificatorul AI, antrenat pe un set de date sintetic, creat manual, recomandă trei stiluri de design interior care se potrivesc cel mai bine utilizatorului. Această componentă aduce valoare prin personalizare și ghidare, reducând semnificativ incertitudinea în procesul de alegere a unui stil decorativ.

Ideea complexă implementată este procesul de generare a imaginilor cu camere redeterminate, pornind de la fotografia încărcată. Aceasta implică utilizarea tehnologiilor ControlNet și Stable Diffusion, două modele de inteligență artificială avansate în procesarea imaginilor. ControlNet permite identificarea structurii camerei și păstrarea elementelor esențiale (precum forma peretilor, ferestre, uși), în timp ce Stable Diffusion generează o imagine nouă, decorată conform stilului și paletei alese. Integrarea acestui flux AI într-o aplicație mobilă a presupus o serie de provocări legate de comunicarea dintre platforme, optimizarea timpului de răspuns și gestionarea imaginilor, dar a oferit rezultate vizuale remarcabile și realiste.

## 1.5 Structura lucrării

Lucrarea este structurată în patru capitole:

1. Introducere - prezentarea temei, a motivației și a scopului lucrării;
2. Preliminarii - descrierea noțiunilor teoretice și a tehnologiilor utilizate;
3. Contribuția proprie - detalii tehnice despre implementarea aplicației;
4. Concluzii - sinteza rezultatelor și posibile direcții viitoare.

## 2. Preliminarii

### 2.1 Noțiuni teoretice fundamentale

#### 2.1.1 Arhitectura client-server

Aplicația *Interio* este construită respectând principiile arhitecturii client–server, model utilizat pe scară largă în dezvoltarea aplicațiilor. În acest model, responsabilitățile sunt distribuite între două componente principale: clientul, componenta care rulează pe dispozitivul utilizatorului (aplicația mobilă), responsabilă cu afișarea informației și preluarea interacțiunilor și serverul, componenta care se ocupă cu logica aplicației, gestionarea datelor și oferirea de răspunsuri la cererile clientului.

Comunicarea dintre cele două se realizează printr-un **API RESTful**, care permite schimbul de date într-un mod standardizat, folosind metodele HTTP (GET, POST, PUT, DELETE) și date într-un format standart (JSON). Astfel, clientul poate solicita, crea, modifica sau șterge resurse, iar serverul răspunde urmând regulile definite în API.

Un aspect important al acestui mod de comunicare este faptul că interacțiunea este *stateless*, ceea ce înseamnă că fiecare cerere trimisă de client trebuie să conțină toate datele necesare pentru a fi procesată. Această abordare permite o scalabilitate mai bună și reduce complexitatea de gestionare a sesiunilor pe server.

#### 2.1.2 Structurarea aplicației pe straturi (*n-tier architecture*)

Arhitectura stratificată, cunoscută și sub denumirea de *n-tier architecture*, reprezintă o metodă de organizare a aplicațiilor complexe prin separarea clară a responsabilităților pe mai multe niveluri logice. În această arhitectură, fiecare strat este responsabil pentru un set specific de funcționalități, iar comunicarea se realizează strict între straturi adiacente. Această abordare favorizează modularitatea, testabilitatea, mențenanța și scalabilitatea aplicației. Straturile principale ale arhitecturii sunt: stratul de prezentare (frontend), stratul de logică de aplicație (servicii și controlere), stratul de acces la date (repository și modele) și stratul de inteligență artificială.

De asemenea, în cadrul aplicației *Interio*, structura adoptată urmează principiile arhitecturii **MVC (Model–View–Controller)**, care separă clar datele, logica de aplicație și interfața cu utilizatorul, facilitând dezvoltarea modulară, reutilizarea codului și o testare mai eficientă a componentelor.

#### 2.1.3 Persistența datelor cu ORM

Pentru gestionarea datelor în cadrul aplicației a fost utilizat un ORM (Object-Relational Mapping), o tehnologie care facilitează interacțiunea între aplicații

scrise într-un limbaj orientat pe obiecte și bazele de date relaționale. Concret, aceasta permite dezvoltatorului să lucreze cu obiecte specifice limbajului de programare al aplicației în locul comenziilor SQL, simplificând semnificativ operațiile de salvare, actualizare sau interogare a datelor.

Un avantaj major oferit de ORM-uri este suportul pentru migrări. Acestea reprezintă o modalitate prin care modificările aduse modelului de date (adăugarea unui câmp sau a unei relații între entități) sunt transpusă automat în modificările schemei bazei de date. Prin comenzi precum ‘*Add-Migration*’ și ‘*Update-Database*’, schema bazei de date poate fi sincronizată cu modelele aplicației, păstrând istoricul modificărilor.[8] Acest proces automatizat de gestionare a evoluției bazei de date contribuie semnificativ la coerența proiectului și la evitarea erorilor ce pot apărea în cazul modificării manuale a tabelelor.

#### 2.1.4 Model de clasificare - Random Forest

Învățarea automată este o ramură a inteligenței artificiale care se ocupă cu dezvoltarea unor algoritmi capabili să învețe modele și relații din date, fără a fi explicit programăți.

Învățarea supervizată presupune existența unui set de date etichetat, unde fiecare instanță are asociată o valoare a variabilei de ieșire. Modelul este antrenat pentru a învăța această relație și apoi utilizat pentru a prezice valorile de ieșire pentru date noi.[10]

Modelele de clasificare sunt tipuri de algoritmi de învățare automată supervizată care au scopul de a prezice o clasă sau categorie pe baza unor date de intrare.

Un arbore de decizie este un model de clasificare exprimat sub forma unei partiționări recursive a spațiului de instanțe. Structura sa este ierarhică și orientată, având un nod rădăcină care reprezintă întregul set de date și mai multe noduri interne care corespund unor condiții logice aplicate asupra unor caracteristici, în funcție de care sunt împărțite datele în subspații distințe. Acest proces continuă până la nodurile terminale (frunzele), care oferă o predicție.[11]

Random Forest este un model de învățare automată de tip *ensemble* (grupare de modele), care combină mai mulți arbori de decizie pentru a obține o predicție mai robustă și mai precisă.[4] Fiecare arbore este antrenat pe un subset aleator de date obținut folosind tehnica *bootstrap*, care presupune un mod de selecție aleatorie (*sampling with replacement*) din setul de antrenament. Astfel, anumite exemple din datele inițiale pot apărea de mai multe ori în același subset, în timp ce altele pot lipsi complet. Această tehnică introduce diversitate între arbori și contribuie la reducerea varianței modelului și la riscului de supraînvățare (*overfitting*), adică modelul nu învăță doar particularitățile setului de antrenament, ci generalizează mai bine pe date necunoscute inițial.[14]

Pentru fiecare subset de date se antrenează un arbore de decizie independent, care învăță reguli proprii de clasificare, adaptate instanțelor pe care le primește. Acest proces este repetat pentru un număr mare de arbori, iar rezultatele sunt agregate prin vot majoritar la clasificare sau prin mediere la regresie.

## 2.1.5 Modele de difuzie pentru generarea imaginilor

O rețea neuronală este un model matematic inspirat de modul în care funcționează creierul uman, format dintr-un număr mare de neuroni artificiali interconectați. Acești neuroni lucrează împreună pentru a învăța tipare complexe din date, aplicând transformări succesive asupra informației primite. Fiecare neuron calculează o valoare numerică numită activare, obținută prin aplicarea unei funcții matematice asupra unei combinații de semnale primite de la alți neuroni. Aceste semnale sunt ponderate cu valori ajustabile numite greutăți (*weights*). Neuronii de intrare preiau date din mediu, iar în funcție de arhitectura rețelei, activările pot fi transmise mai departe sau pot produce o ieșire finală, utilizată pentru luarea unei decizii sau realizarea unei acțiuni.[12]

Un model generativ este un tip de rețea neuronală conceput pentru a învăța distribuția probabilistică a unui set de date și apoi pentru a genera noi exemple originale și realiste din aceeași distribuție.[7]

Modelele de difuzie sunt o clasă de modele generative probabilistice care funcționează prin învățarea unui proces invers, de reducere a zgomotului dintr-o imagine. Procesul începe prin adăugarea treptată de zgomot aleatoriu la o imagine (*forward process*), până când aceasta devine complet aleatoare. Modelul este apoi antrenat să învețe cum să reconstruască imaginea originală pas cu pas, eliminând zgomotul (*reverse process*). După antrenare, modelul poate genera, pornind de la zgomot pur, o imagine nouă și realistă, apropiată de distribuția imaginilor din setul de antrenament.

**Stable Diffusion** este un exemplu de model generativ de tip difuzie, antrenat să creeze imagini fotorealiste pornind de la o descriere textuală.

## 2.1.6 Control Net

Una dintre limitările modelelor de difuzie este lipsa controlului asupra structurii imaginii generate. De exemplu, pentru a păstra forma unei camere sau pentru a redecora o imagine reală, este dificil să obținem rezultate coerente doar cu un prompt text.

ControlNet este o structură de rețea neuronală care adaugă un canal suplimentar de condiționare peste un model de difuzie existent.[15] Acest canal permite modelului să țină cont nu doar de promptul text, ci și de informații vizuale suplimentare, precum hărți de contur, hărți de adâncime, segmente semantice sau structuri geometrice.

## 2.2 Tehnologii utilizate

### 2.2.1 Frontend - React Native

Pentru dezvoltarea interfeței aplicației *Interio*, a fost utilizat un set de tehnologii moderne care susțin o abordare modulară, scalabilă și compatibilă cu platformele mobile Android și iOS. Alegerea acestor tehnologii a fost motivată de nevoia de a obține o

aplicație cu aspect profesional, ușor de întreținut și cu o experiență fluidă și intuitivă pentru utilizator.

Framework-ul principal utilizat pentru dezvoltarea aplicației este **React Native**, o soluție *open-source* creată de Meta, care permite dezvoltarea aplicațiilor mobile *cross-platform*, folosind un singur cod sursă scris în *JavaScript* sau *TypeScript*.

Pentru a accelera dezvoltarea și testarea aplicației, a fost utilizată platforma **Expo**. Aceasta oferă o suită de unele și servicii utile, precum aplicația *Expo Go*, pentru testare instantă pe dispozitive reale, posibilitatea de a genera build-uri fără configurații complexe și integrarea simplificată a modulelor native (*expo-router*, *expo-secure-store*).

Pentru asigurarea unei interfețe moderne și coerente pe tot parcursul aplicației, a fost folosită biblioteca de componente **UI Kitten**. Aceasta oferă o gamă variată de componente predefinite, precum butoane, liste sau formulare, care pot fi personalizate tematic și integrate rapid în aplicație. Alegerea UI Kitten s-a bazat pe documentația clară și exemplele intuitive, care au simplificat procesul de integrare în aplicație.[13]

Comunicarea cu API-urile expuse de backend a fost realizată cu ajutorul bibliotecii **Axios**, o soluție pentru trimitera cererilor HTTP și gestionarea răspunsurilor asincrone. În cadrul aplicației, Axios a fost configurat într-un fișier dedicat pentru a centraliza toate setările legate de conexiunea cu API-urile *Interio*. A fost alesă această bibliotecă datorită compatibilității bune cu React Native, utilizării intuitive și a documentației clare.

## 2.2.2 Backend - ASP.NET Core

Pentru dezvoltarea componentei server-side a aplicației *Interio*, a fost utilizat framework-ul **ASP.NET Core**, o platformă *open-source* și *cross-platform* dezvoltată de Microsoft.[1] Limbajul de programare folosit în acest context este **C#**, cunoscut pentru sintaxa sa clară, suportul extins pentru programarea orientată pe obiecte și integrarea nativă cu ASP.NET. Alegerea acestui limbaj s-a bazat în principal pe experiența anterioară în utilizarea sa.

Pentru autentificare și autorizare, aplicația utilizează tehnologia **JWT (JSON Web Tokens)**, un standard deschis (RFC 7519) care permite transmiterea securizată a informațiilor între client și server sub forma unui obiect JSON.[5] JWT este folosit pentru a implementa o arhitectură *stateless*, în care serverul nu trebuie să păstreze sesiuni active în memorie, toate informațiile relevante despre utilizator fiind conținute în tokenul transmis de client în fiecare cerere. Un token JWT este compus din trei părți: (1) antetul, care specifică algoritmul de semnare (ex: HS256 sau RS256); (2) conținutul, care include informații precum utilizatorul, rolurile și data expirării; și (3) semnatura digitală, generată pe baza celorlalte două părți și a unei chei secrete, pentru a garanta integritatea tokenului.

Pentru documentarea și testarea endpoint-urilor API, a fost integrat **Swagger**, prin pachetul *Swashbuckle.AspNetCore*. Swagger generează automat o interfață web unde pot fi vizualizate toate ruturile disponibile, fiind util atât în etapa de dezvoltare, cât și cea de testare.

Tehnologia ORM folosită în ASP.NET Core este **Entity Framework**, un instrument dezvoltat de Microsoft care permite maparea claselor C# la tabelele corespunzătoare din baza de date, fără a fi necesară scrierea explicită a interogărilor SQL.

### 2.2.3 Baza de date - MSSQL

Pentru stocarea datelor aplicației *Interio* a fost utilizat **Microsoft SQL Server (MSSQL)**, un sistem de gestiune a bazelor de date relaționale dezvoltat de Microsoft.[9] MSSQL oferă suport complet pentru tranzacții, relații între tabele, integritate referențială și funcții avansate de interogare, adesea folosit pentru proiecte construite cu ASP.NET.

### 2.2.4 Tehnologii AI

**Scikit-learn** este o bibliotecă *open-source* pentru Python, care oferă un set extins de algoritmi de învățare automată. În aplicație, această bibliotecă a fost folosită pentru a testa și compara mai mulți clasificatori, printre care Random Forest, Support Vector Machines (SVM) și regresia logistică.

Modelul de bază utilizat pentru generarea propriu-zisă a imaginilor este *runwayml/stable-diffusion-v1-5*, selectat în urma unor teste comparative datorită echilibrului pe care îl oferă între calitatea vizuală și fidelitatea față de instrucțiunile din prompt. Pentru a păstra conturul camerei reale, este utilizată rețeaua *lllyasviel/control\_v11p\_sd15\_mlsd*, o versiune ControlNet specializată în lucrul cu hărți **MLSD (Multi-Line Segment Detector)**. Aceste hărți sunt generate automat cu ajutorul algoritmului auxiliar *MLSDdetector*, care analizează imaginea camerei încărcate de utilizator și extrage liniile structurale predominante (muchiile peretilor sau ale obiectelor de mobilier) oferind astfel o reprezentare geometrică precisă. Pentru a accelera procesul de generare, sistemul utilizează *UniPCMultistepScheduler*, un algoritm de planificare multi-pas care optimizează inferența în rețeaua de difuzie, reducând timpul de procesare fără a compromite calitatea rezultatului.

De asemenea, biblioteca **torch** este esențială în manipularea și rularea modelelor, fiind utilizată și pentru asigurarea reproductibilității rezultatelor prin fixarea semintei aleatorii cu funcția *torch.manual\_seed()*.

Imaginea finală este redimensionată folosind un algoritm **LANCZOS**, pentru a fi afișată clar și eficient în aplicația mobilă, într-un format comparativ cu fotografia originală.

### 2.2.5 Alte tehnologii și instrumente auxiliare

**Flask** este un micro-framework pentru dezvoltarea aplicațiilor web în Python, cunoscut pentru simplitatea și flexibilitatea sa. A fost utilizat pentru a construi un API care gestionează comunicarea dintre aplicația mobilă și funcțiile Python care implementează modelele AI.[3]

Pentru gestionarea codului sursă a fost utilizat sistemul de control al versiunilor **Git**, împreună cu platforma GitHub.

În ceea ce privește mediile de dezvoltare, au fost folosite două editoare principale: **Visual Studio Code** pentru partea de frontend (React Native) și **JetBrains Rider** pentru partea de backend (ASP.NET Core). Ambele oferă suport avansat pentru limbajele și framework-urile utilizate, extensii utile și integrare cu Git.

Pentru organizarea sarcinilor de lucru, stabilirea priorităților și urmărirea progresului proiectului, a fost folosită aplicația **Trello**. Task-urile au fost împărțite în categorii specifice fluxului de lucru (*to do, in progress, done*) și etichetate în funcție de prioritate (urgent, mediu, scăzut), ceea ce a permis o vizualizare clară a gradului de importanță al fiecărei activități, contribuind la planificarea mai eficientă a timpului.

## 2.3 Contextul actual

În ultimii ani, domeniul aplicațiilor de design interior asistat de inteligență artificială a cunoscut o dezvoltare rapidă datorită progreselor în învățarea automată și în modelele generative de imagini.

*Home AI* sau *Reimagine Home*, de exemplu, permit încărcarea unei fotografii și alegerea unui stil și a unei palete de culori dintr-o listă predefinită, dar nu oferă posibilitatea introducerii unor preferințe detaliate (ex: mobilier, decorațiuni) și nici un mecanism de recomandare personalizată.

*Planner 5D* este o aplicație web care oferă unelte avansate de proiectare 3D și simulare, dar procesul este în mare parte dificil și consumator de timp. Aceasta oferă un editor vizual care nu este intuitiv și nu permite construirea rapidă a unei schițe pentru camere goale fără a avea experiență tehnică.

În procesul de cercetare a celor mai eficiente metode de generare a imaginilor pentru amenajări interioare, am analizat mai multe modele generative, precum și soluții propuse în literatura de specialitate. Un exemplu remarcabil este lucrarea lui Junming Chen [6], care propune un sistem bazat pe *Stable Diffusion* extins cu o rețea personalizată de control, numită *Interior Design Control Network (IDCN)*. Acest model permite generarea de schițe și randări interioare care păstrează fidel structura camerei inițiale, pe baza unei imagini fără mobilier, folosind un set de date special construit (IFAPD) și o funcție compusă de pierdere pentru antrenare. Rezultatele obținute în lucrare arată performanțe superioare în ceea ce privește realismul, coerența funcțională și varietatea stilistică, comparativ cu alte modele generative existente.

Totuși, IDCN presupune construirea și adnotarea unui set complex de date (IFAPD), antrenarea unei rețele aditionale și integrarea într-un flux de lucru dedicat, ceea ce poate limita fezabilitatea implementării într-o aplicație mobilă, unde timpul de inferență, resursele hardware și scalabilitatea sunt critice.

În analiza comparativă, am luat în considerare și alte modele generative populare,

precum Midjourney și DALL·E 2. Deși ambele pot genera imagini de înaltă calitate vizuală, acestea nu oferă un control precis asupra structurii spațiului interior. Midjourney, de exemplu, excelează în generarea de compozиții artistice, dar nu menține coerенță geometrică a imaginilor de intrare. DALL·E 2 nu permite condiționarea generării pe baza unei imagini de intrare, ceea ce îl face inadecvat pentru scenarii în care structura fizică a camerei trebuie menținută. În plus, am testat și varianta standard de *Stable Diffusion*, fără rețea de control, dar rezultatele au fost instabile, cu mobilă generată în poziții incoerente sau în afara contextului structural.

## 2.4 Obiective generale ale lucrării

Scopul principal al acestei lucrări este dezvoltarea unei aplicații mobile, care să faciliteze procesul de amenajare interioară prin recomandări personalizate și generarea de vizualizări realiste ale spațiilor decorate. Într-un context în care utilizatorii sunt adesea confruntați cu un număr mare de opțiuni și cu dificultatea de a alege un stil coherent, aplicația *Interio* vine în sprijinul acestora prin integrarea tehnologiilor de inteligență artificială pentru procesarea imaginilor.

Un prim obiectiv urmărit este realizarea unui sistem intuitiv de recomandare, capabil să identifice preferințele estetice, stilul de viață și trăsăturile de personalitate ale utilizatorului, printr-un chestionar. Pe baza răspunsurilor, un algoritm de clasificare va sugera trei stiluri de design interior care se potrivesc cel mai bine profilului identificat, alături de paletele de culori asociate. Astfel, se creează o experiență adaptată fiecărui utilizator.

Un al doilea obiectiv esențial constă în dezvoltarea unei funcționalități capabile să genereze automat o imagine realistă cu o cameră redescăpată, pornind de la o fotografie încărcată de utilizator. Această funcționalitate este susținută de modele AI de tip ControlNet și Stable Diffusion, care permit simularea vizuală a decorului ales, ținând cont de tipul camerei, stilul și culorile preferate.

În completarea acestor funcționalități, un alt obiectiv important vizează implementarea unui editor vizual interactiv, prin intermediul căruia utilizatorii pot configura manual aspectul unei camere, începând de la o cameră goală. Aceștia pot adăuga și poziționa elemente de mobilier dintr-un catalog predefinit, creând astfel o reprezentare schematică ce reflectă propriile idei de amenajare. Această abordare completează posibilitatea de generare automată a mobilierului și decorului, făcând experiența mai versatilă și mai apropiată de nevoile reale ale utilizatorului.

Prin atingerea acestor obiective, lucrarea își propune să demonstreze fezabilitatea unei soluții care combină personalizarea, interactivitatea și tehnologia modernă într-un domeniu cu aplicabilitate practică ridicată (designul interior). Aplicația *Interio* răspunde unei nevoi reale de claritate vizuală și asistență personalizată în luarea deciziilor legate de amenajarea propriei locuințe.

# 3. Contribuția proprie

## 3.1 Arhitectura aplicației

În figura 3.1 este reprezentată atât arhitectura generală client-server a aplicației *Interio*, cât și structurarea acesteia pe mai multe straturi logice.

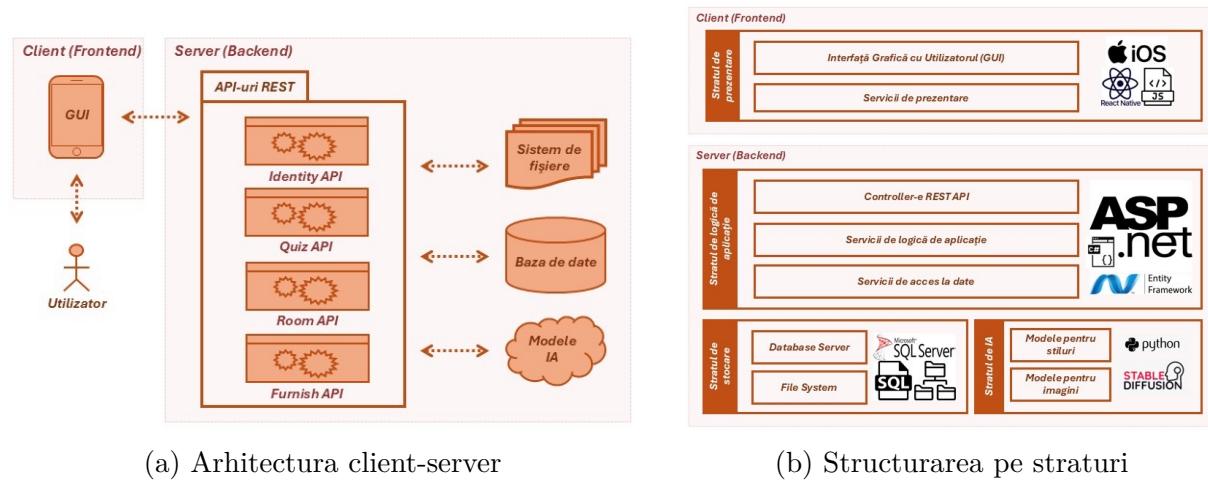


Figura 3.1: Arhitectura aplicației

Stratul de prezentare este reprezentat de aplicația mobilă dezvoltată în React Native, responsabilă pentru interacțiunea directă cu utilizatorul. Aceasta include interfață grafică (GUI) și serviciile de prezentare care preiau și afișează datele primite de la backend, oferind o experiență fluidă și intuitivă, atât pe platforme iOS, cât și Android.

Stratul de logică de aplicație este dezvoltat folosind ASP.NET Core, este localizat în componenta backend și este organizat la rândul său în două substraturi: unul dedicat API-urilor REST, care conține controlerelor ce gestionează cererile HTTP și rutele publice ale aplicației și unul dedicat serviciilor, care conțin regulile de validare și fluxurile funcționale ale aplicației.

Stratul de stocare a datelor este responsabil pentru persistența informațiilor în baza de date și fișiere. Acesta este organizat în două substraturi: substratul de acces la date, unde repository-urile oferă o abstractizare peste operațiile cu baza de date folosind Entity Framework Core și substratul de modele de entități, care definește clasele corespunzătoare tabelelor din baza de date. Structura entităților și relațiilor dintre ele este ilustrată în Diagrama Entitate-Relație (*Entity-Relationship Diagram*) prezentată în Anexa 3, Figura 7.

Stratul de inteligență artificială este implementat ca un modul separat în Python și expus printr-un API folosind framework-ul Flask. Acest modul implementează două funcționalități bazate pe modele AI: cea pentru clasificarea stilurilor, bazată pe Random Forest, folosită pentru a determina preferințele utilizatorului pe baza răspunsurilor din quiz și cea pentru generarea imaginilor, bazată pe Stable Diffusion și ControlNet, utilizată

pentru a crea variante redecorate ale camerelor încărcate.

Comunicarea între aceste straturi se realizează prin intermediul unor interfețe bine definite, respectând principiile arhitecturii în straturi. Stratul de prezentare (frontend-ul în React Native) interacționează cu stratul de logică de aplicație prin cereri HTTP către API-urile REST expuse de backend. Backend-ul procesează aceste cereri în controllere, care transmit datele mai departe către serviciile interne pentru validare, procesare și luarea deciziilor. Serviciile backend colaborează cu stratul de stocare a datelor prin intermediul repository-urilor, care realizează operații de citire și scriere în baza de date utilizând Entity Framework Core. În paralel, atunci când sunt necesare predicții de stil sau generări de imagini, backend-ul comunică cu stratul de inteligență artificială prin apeluri HTTP către API-ul Flask, trimițând datele relevante și așteptând răspunsuri.

Această separare clară a responsabilităților contribuie la o arhitectură modulară, scalabilă și ușor de întreținut, facilitând dezvoltarea și extinderea ulterioară a aplicației.

## 3.2 UI și UX

În dezvoltarea aplicației *Interio*, experiența utilizatorului (UX) și interfața utilizator (UI) au fost tratate cu o atenție deosebită, pentru a oferi utilizatorilor o experiență cât mai plăcută, intuitivă și personalizată. Design-ul vizual urmărește un stil modern și fluent, folosind componente create de mine sau oferite de framework-ul UI Kitten, care asigură consistența elementelor grafice.

Aplicația pune accent pe o experiență intuitivă, ghidând utilizatorii pas cu pas în utilizarea funcționalităților principale. De exemplu, imediat după prima autentificare, utilizatorul este direcționat către un chestionar, care determină stilul de design interior și paleta de culori care i se potrivește și apoi le afișează pe profilul acestuia. Acest aspect aduce un plus de interes, întrucât utilizatorul simte că aplicația se adaptează gusturilor sale.

Feedback-ul vizual este prezent pentru toate acțiunile importante: confirmări la trimiterea chestionarului, mesaje informative la ștergerea camerelor sau salvarea modificărilor, precum și afișarea unui indicator de încărcare în procesele mai lente. De asemenea, în lipsa datelor (ex: nicio cameră încărcată) sunt afișate mesaje explicite, evitând confuziile.

Designul aplicației este adaptat pentru diferite dimensiuni de ecran și comportamente ale platformelor mobile. Folosirea componentelor *SafeAreaView* și *ScrollView* permite o navigare confortabilă pe ecrane de orice dimensiune, iar *KeyboardAvoidingView* asigură că introducerea textelor nu este obstrucționată de tastatură pe dispozitive iOS. De asemenea, interfața respectă principii minime de accesibilitate, asigurând un contrast bun între text și fundal, fonturi lizibile și spațiere adecvată, care contribuie la o experiență vizuală clară.

Prin toate aceste decizii de design, *Interio* reușește să combine estetica vizuală cu funcționalitatea, oferind o aplicație ușor de utilizat și adaptată nevoilor utilizatorilor.

## 3.3 Autentificare

### 3.3.1 Pagina de Sign up și Login

Aplicația *Interio* oferă o funcționalitate completă de autentificare și gestionare a conturilor de utilizator, construită pe baza pachetului *ASP.NET Identity*[2] care aderă la cele mai bune practici de securitate. Aceasta include înregistrare, autentificarea și validarea token-urilor de acces. Comunicarea între aplicația mobilă și server se face printr-o arhitectură RESTful, utilizând token-uri JWT pentru securizarea sesiunilor. Imagini ilustrative cu interfața pentru înregistrare și autentificare pot fi consultate în Figura 1 din Anexa 1.

Pentru înregistrare (sign up), utilizatorul introduce o adresă de e-mail și o parolă care trebuie să respecte o serie de reguli de securitate (minim 6 caractere, cel puțin o literă mare, un caracter special, o cifră și confirmarea corectă a parolei). Parolele utilizatorilor nu sunt stocate în clar, ci sunt procesate automat de *ASP.NET Identity*, care aplică algoritmul *PBKDF2* împreună cu un *salt* unic per utilizator. Doar hash-ul rezultat este salvat în baza de date, în timp ce parola inițială nu este niciodată păstrată sau transmisă. De asemenea, se verifică ca e-mailul să aibă o formă validă și să nu fie deja asociat unui alt cont. După validare, contul este salvat în baza de date cu un rol implicit de “USER”.

La autentificare, utilizatorul introduce adresa de e-mail și parola. Dacă datele sunt corecte, serverul generează un Access Token JWT valabil o oră, care conține ID-ul, e-mailul și rolurile utilizatorului și un Refresh Token salvat în baza de date, folosit pentru reînnoirea sesiunii.

Token-urile sunt salvate securizat cu *expo-secure-store*. Dacă utilizatorul nu a completat quiz-ul de personalitate, acesta este redirectionat către pagina de quiz, altfel, acesta este directat către pagina principală (*home page*).

În plus, interfața oferă validări în timp real, feedback vizual în caz de eroare (ex: mesaje de eroare că parola nu respectă cerințele), precum și redirectionări între ecranele Sign Up și Login, contribuind astfel la o experiență de utilizare clară și intuitivă.

### 3.3.2 Resetarea parolei

Funcționalitatea de resetare a parolei le oferă utilizatorilor posibilitatea de a-și recupera accesul la cont într-un mod securizat, folosind adresa de e-mail personală și un cod de verificare temporar. Imagini ilustrative cu interfața pentru procesul de resetare a parolei pot fi consultate în Figura 2 din Anexa 1. După introducerea unei adrese valide în aplicație, serverul verifică dacă aceasta este asociată unui cont existent. Dacă da, se generează un cod numeric aleatoriu de 6 cifre, valabil timp de 10 minute, care este trimis prin e-mail și stocat temporar într-o structură *ConcurrentDictionary* din clasa statică *TemporaryResetStore*. Acest cod acționează ca un mecanism de autentificare temporară și este valabil o singură dată. După expirarea sau utilizarea sa, codul este invalidat

automat, reducând riscul de acces neautorizat.

Avantajul utilizării unei structuri *in-memory* constă în evitarea stocării codurilor în baza de date, ceea ce simplifică implementarea și oferă o soluție rapidă și volatilă, potrivită pentru coduri sensibile cu durată de viață scurtă.

Pentru trimiterea codului de verificare prin e-mail au fost analizate două soluții: integrarea cu un API extern, **Resend**, și utilizarea unui server **SMTP**. Deși Resend a fost testat inițial, planul gratuit permite trimiterea de mesaje doar către o adresă de test, limitând funcționalitatea. În cele din urmă, a fost adoptată o soluție SMTP, configurată cu un cont Gmail și o parolă de aplicație obținută prin autentificare în doi pași oferit de Google. Această abordare s-a dovedit stabilă și suficientă pentru cerințele aplicației, în plus fiind și gratuită.

## 3.4 Chestionarul de personalitate și preferințe

### 3.4.1 Scopul și implementarea chestionarului

Chestionarul este un element esențial în aplicația *Interio*, fiind creat pentru a identifica stilul de design interior potrivit fiecărui utilizator. După prima autentificare, utilizatorul este direcționat către pagina cu chestionarul, pe care trebuie să îl completeze pentru a putea accesa restul funcționalităților aplicației. După completare, răspunsurile sunt validate și convertite în format numeric (A–0, B–1, C–2, D–3), urmând a fi trimise către backend.

Pentru clasificarea stilului de design interior preferat, au fost definite șase stiluri distințe și populare: *Minimalist* (centrat pe simplitate și funcționalitate, preferat de cei care caută ordine și liniște), *Boho* (un stil artistic și relaxat, inspirat din natură, ideal pentru personalități nonconformiste), *Rustic* (cald și autentic, cu influențe rurale, potrivit celor care apreciază confortul și tradiția), *Eclectic* (creativ și expresiv, specific celor cu spirit artistic) *Glam* (rafinat și luxos, cu materiale elegante, preferat de persoanele carismatice și sociabile) și *Industrial* (caracterizat de un aspect urban și funcțional, apreciat pentru designul său practic și nefinisat).

Chestionarul este alcătuit dintr-o serie de întrebări structurate în două categorii: *ChoiceText* (răspunsuri sub formă de text) și *ChoiceImage* (răspunsuri sub formă de imagini ilustrative).

Am urmărit ca fiecare întrebare să scoată în evidență aspecte relevante pentru alegerea unui stil de design interior, cum ar fi modul în care cineva preferă să-și petreacă timpul liber, tipul de atmosferă pe care îl caută pentru casa sa sau preferințele legate de culori, materiale și texturi. Fiecărei opțiuni de răspuns i-au fost asociate unul sau mai multe stiluri relevante, în funcție de semnificația lor. De exemplu, la întrebarea „Cum arată ziua ta liberă ideală?”, opțiunea „O zi relaxantă acasă cu o carte” a fost asociată cu stilul Minimalist, „Participarea la un concert sau o galerie de artă” a fost atribuită stilului Eclectic, în timp ce „Drumeții și relaxare în natură” reflectă preferințe tipice pentru stilul

Boho sau Rustic. O opțiune precum „Organizarea unei cine festive pentru prieteni” a fost asociată cu stilurile Glam și Industrial.

### 3.4.2 Crearea dataset-ului de antrenare

Am construit manual setul de antrenare în format tabelar, folosind un fișier Excel. Am definit un set ideal de răspunsuri, considerat reprezentativ pentru fiecare stil. Apoi, am generat 20 variații pentru fiecare stil, modificând unul, două sau trei răspunsuri, pentru a introduce diversitate în datele de antrenament. Această strategie permite modelului să învețe nu doar cazurile ideale, ci și variațiile naturale pentru utilizatorii cu preferințe similare.

Pentru a extinde setul de date de antrenare și pentru a obține exemple mai apropiate de realitate, am creat o versiune interactivă a chestionarului, care a fost distribuită prietenilor și familiei. Scopul a fost de a colecta alte variante de răspuns, dincolo de scenariile ideale create manual. Am inclus la finalul chestionarului următoarea întrebare: “Care dintre stiluri consideri că te reprezintă cel mai bine?”, pentru a putea asocia fiecărui set de răspunsuri o etichetă aleasă direct. Aceste exemple au fost ulterior adăugate în setul de date și au contribuit la îmbunătățirea acurateței clasificatorului, oferindu-i o mai bună capacitate de generalizare asupra unor răspunsuri noi.

Structura finală a dataset-ului este compusă din exemple, fiecare conținând câte 8 răspunsuri (câte unul pentru fiecare întrebare) și o etichetă corespunzătoare stilului de design.

### 3.4.3 Implementarea clasificatorului de stiluri

Pentru a oferi recomandări personalizate de design interior, aplicația *Interio* integrează un clasificator AI ce determină stilurile preferate ale utilizatorului pe baza răspunsurilor oferite în chestionar.

Modelul a fost implementat în Python folosind biblioteca scikit-learn și a fost antrenat cu un Random Forest Classifier cu 100 de arbori de decizie. Aceasta oferă o acuratețe bună în clasificarea pe baza datelor tabulare discrete, fiind robust și ușor de interpretat. O secvență de cod este disponibilă în Figura 6 din Anexa 2.

Pentru alegerea celui mai potrivit clasificator, au fost testați mai mulți algoritmi clasici de învățare automată: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Tree Classifier și Logistic Regression. Deși fiecare model a fost antrenat și evaluat folosind același set de date, aceștia au înregistrat o acuratețe mai scăzută. KNN - 80%, SVM - 80%, Decision Tree Classifier - 68%, Logistic Regression - 84%.

Modelul care a oferit cele mai bune rezultate a fost Random Forest, antrenat pe un set de 150 de exemple și testat pe un set de 25. Acesta a obținut o acuratețe de 96%, cu valori ridicate ale metricilor de performanță (*precision*, *recall* și *f1-score*), așa cum se poate observa în Figura 3.2, pentru majoritatea claselor. Rezultatele indică o bună

capacitate de generalizare a modelului și o distribuție echilibrată a predicțiilor, în special în condițiile unui set de date construit manual.

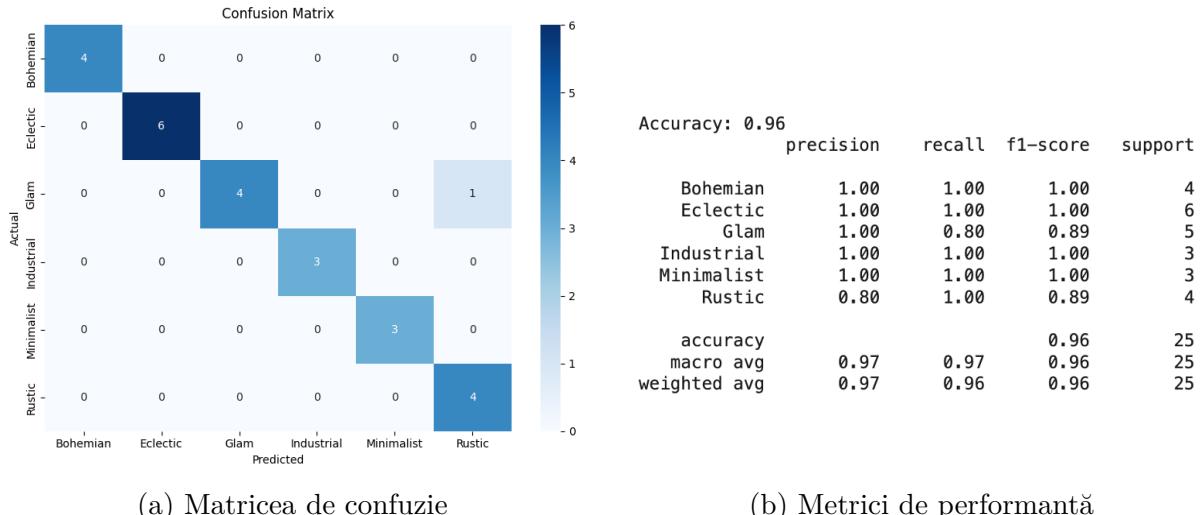


Figura 3.2: Evaluarea clasificatorului Random Forest

Așadar, a fost ales Random Forest, deoarece oferă un echilibru foarte bun între precizie, viteză de antrenare și interpretabilitate și nu necesită scalarea datelor sau preprocesări complexe. De asemenea, prin agregarea rezultatelor mai multor arbori de decizie antrenați pe subseturi diferite ale datelor, Random Forest reduce riscul de supraînvățare (*overfitting*) și reușește să generalizeze mai bine pe date noi. Aceste caracteristici l-au transformat într-o alegere practică și eficientă pentru clasificarea stilurilor de design interior în aplicația *Interio*.

După antrenare, clasificatorul a fost expus printr-un API Flask ce primește, printr-un endpoint POST (/predict-style), un set de opt răspunsuri codificate numeric (A=0, B=1, C=2, D=3) și returnează o listă cu cele mai probabile trei stiluri asociate.

API-ul de AI este apelat din stratul logicii de aplicație, implementat în .NET, imediat după ce utilizatorul finalizează chestionarul. Răspunsurile sunt salvate în baza de date, prelucrate și trimise către API-ul Flask. Aceasta returnează cele trei stiluri de design interior care se potrivesc cel mai bine profilului utilizatorului, ordonate în funcție de scorul de încredere generat de model. Stilurile recomandate sunt apoi salvate în tabela UserPreferences, alături de paleta de culori asociată fiecărui.

Această arhitectură modulară permite actualizarea sau înlocuirea modelului de clasificare fără impact asupra restului aplicației și oferă o separare clară între stratul logicii de aplicație și cel de AI.

## 3.5 Pagina principală și de profil

Pagina acasă este punctul central al aplicației *Interio* pentru utilizatorii autentificați, oferind acces rapid la toate camerele deja procesate de utilizator. În cazul în care nu a

fost realizată nicio cameră, este afișat un mesaj sugestiv “No rooms uploaded yet”. În schimb, dacă există camere, acestea sunt afișate într-un ScrollView, fiecare cameră fiind reprezentată de un container care conține imaginea generată de AI, detalii precum tipul camerei, stilul și paleta de culori folosite și un buton de ștergere.

Aplicația *Interio* oferă utilizatorilor autenticați posibilitatea de a-și gestiona contul într-un mod intuitiv. Imagini ilustrative cu interfața pentru gestionarea profilului utilizatorului pot fi consultate în Figura 2 din Anexa 1.

Utilizatorii pot accesa o pagină dedicată profilului, unde își pot vizualiza poza de profil, numele, stilul recomandat și paleta de culori asociată. Sub aceste informații se află o listă de butoane care oferă acces la următoarele funcționalități: posibilitatea de a edita profilul (actualizarea numelui și schimbarea imaginii de profil, imaginea fiind încărcată printr-un formular multipart/form-data și salvată în directorul wwwroot/profile-images, cu linkul corespunzător stocat în baza de date), de a schimba parola (prin validarea parolei vechi și înlocuirea cu una nouă, folosind metodele securizate oferite de ASP.NET Identity), de a reface testul de personalitate (pentru actualizarea preferințelor de design), de a vizualiza detalii despre cele trei stiluri recomandate de clasificator (inclusiv descrieri, trăsături cheie și palete de culori), de a se deconecta (ștergând token-urile de acces și revenind la ecranul de start), respectiv de a șterge definitiv contul (operăție care elimină utilizatorul din baza de date și imaginea asociată, cu un pas suplimentar de confirmare pentru prevenirea ștergerilor accidentale).

### 3.6 Decorarea unei camere

În cadrul aplicației *Interio* a fost implementat un sistem de generare a imaginilor care simulează reamenajarea realistă a unei camere pe baza stilului de design ales din cele recomandate de clasificator.

Procesul de generare a unei imagini redecorate în aplicația *Interio* începe cu un prim pas esențial: alegerea imaginii de start. Figura 3.3 ilustrează vizual cele două opțiuni. Utilizatorii pot opta fie pentru încărcarea unei fotografii ale unei camere reale, fie pentru construirea unei schițe manuale într-un editor vizual interactiv. A doua variantă este utilă în special în situațiile în care se dorește decorarea unei locuințe noi, nu există o fotografie corespunzătoare sau se dorește simularea unei configurații ipotetice. Editorul permite selectarea unui fundal predefinit (o cameră goală) și adăugarea de obiecte de mobilier standard, ce pot fi poziționate și aranjate în mod intuitiv. Soluția s-a dovedit eficientă în contextul în care modelele generative oferă rezultate limitate atunci când sunt condiționate exclusiv pe imagini complet goale.

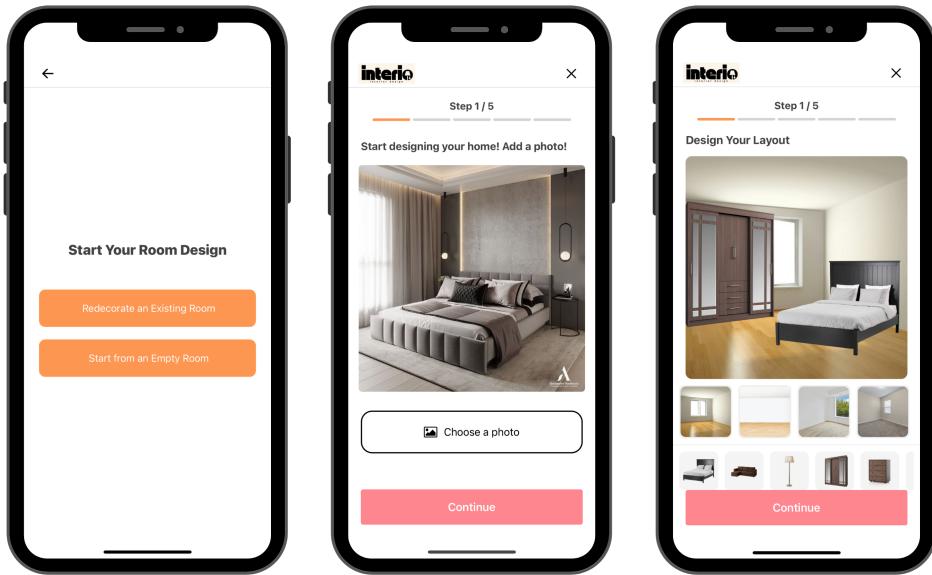


Figura 3.3: Interfața pentru alegerea imaginii de start (fotografie reală sau schită)

După stabilirea imaginii inițiale, aplicația parcurge un flux comun format din patru pași: selectarea tipului camerei, alegerea unui stil de design interior din cele recomandate de clasificator, selectarea unei palete de culori și introducerea optională a unui detaliu personalizat în promptul de generare (de exemplu: „Adaugă o plantă.”). Imaginea finală este generată de modelul AI, afișată într-un format comparativ cu cea inițială și salvată automat în lista personală de camere. Figura 3.4 ilustrează vizual acest flux de UI.

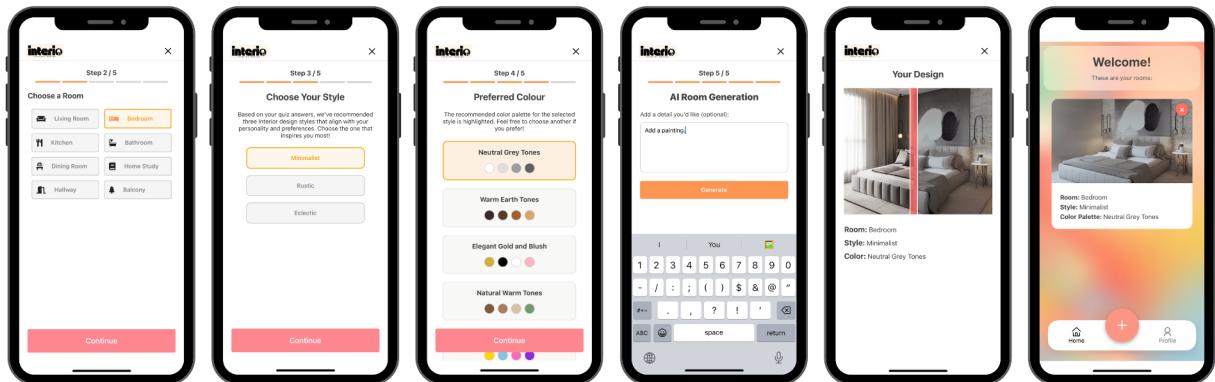


Figura 3.4: Pașii următori în generarea imaginii cu camera redecorată

Tot fluxul UI al funcționalității de decorare a unei camere se poate vedea mai clar în Figura 4, din Anexa 1.

Pentru atingerea acestui obiectiv, a fost implementată în stratul de AI o funcție Python care integrează modelul Stable Diffusion, extins cu ControlNet. Scopul acestei funcții este de a genera o imagine realistă a unei camere reamenajate, ținând cont atât de structura camerei originale, cât și de promptul textual asociat stilului de design ales. O secvență de cod este disponibilă în Figura 6 din Anexa 2.

Imaginea este procesată inițial de algoritmul MLSD (Multi-Line Segment Detector), care extrage liniile structurale predominante ale camerei. Această imagine de tip contur

este ulterior folosită ca intrare de control pentru rețeaua ControlNet.

Pentru generarea imaginii, este utilizat modelul *control\_v11p\_sd15\_mlsd* conectat la versiunea *stable-diffusion-v1-5* prin intermediul unui pipeline oferit de biblioteca *diffusers*. Au fost configurați parametri esențiali pentru generare, precum *guidance\_scale* (care controlează fidelitatea față de prompt) și *num\_inference\_steps* (care influențează calitatea rezultatului), precum și un seed fix (*torch.manual\_seed(0)*) pentru obținerea de rezultate reproducibile. Imaginea generată este redimensionată cu ajutorul unei funcții care aplică un algoritm de tip LANCZOS, oferind astfel o versiune clară și pregătită pentru afișare în aplicația mobilă.

În faza de integrare a modelelor AI în aplicație, au fost întâmpinate dificultăți legate de rularea locală a modelelor de difuzie pe un sistem fără suport GPU dedicat. Inițial, procesul de generare a imaginilor era semnificativ încetinit din cauza lipsei compatibilității cu CUDA pe laptopul utilizat pentru dezvoltare. Pentru a asigura o performanță adecvată, a fost necesară trecerea la un calculator echipat cu placă grafică NVIDIA și suport pentru CUDA, care a permis accelerarea procesării prin GPU. Chiar și în acest context, pentru optimizarea performanței, a fost activată opțiunea *enable\_model\_cpu\_offload*, care reduce consumul de memorie pe GPU în timpul generării.

Funcționalitatea este complet integrată cu backend-ul aplicației printr-un API REST expus cu ajutorul framework-ului Flask, permitând trimitera promptului și a imaginii inițiale de către clientul mobil, precum și recepționarea imaginii generate ca răspuns.

În etapa de explorare a soluțiilor pentru generarea de imagini personalizate, au fost testate mai multe modele generative disponibile. Printre acestea s-a numărat și DALL·E 2, dezvoltat de OpenAI, un model capabil să creeze imagini de înaltă calitate pe baza unor prompt-uri text. Cu toate acestea, această opțiune nu s-a potrivit cerințelor aplicației, deoarece nu permite bazarea pe o imagine de intrare, ci generează de fiecare dată o compoziție complet nouă, fără legătură cu structura camerei inițiale.

Ulterior, atenția a fost concentrată pe modelele din familia Stable Diffusion, care oferă posibilitatea de a fi extinse cu structuri auxiliare precum ControlNet, pentru o generare controlată. În cadrul experimentelor, au fost testate mai multe versiuni ale modelului de bază, printre care *stable-diffusion-v1-4* și *stable-diffusion-2-1*, însă cea mai bună calitate a fost obținută cu modelul *stable-diffusion-v1-5*, care a fost adoptat ca versiune finală în aplicația *Interio* datorită echilibrului optim între detaliu vizual și fidelitatea față de prompt.

În ceea ce privește integrarea imaginii inițiale în procesul de generare, au fost explorate mai multe metode de procesare vizuală, specifice rețelei ControlNet. Printre acestea s-au numărat Canny Edge Detector, Depth Map și MLSD (Multi-Line Segment Detector).

Figura 3.5 ilustrează aceste metode prin perechi de imagini (input/output) pentru același exemplu de cameră și pentru prompt-ul "A Scandinavian-style bed room with minimalist furniture, soft natural light and a neutral modern color palette". Imaginea de

input este cea reprezentată în Figura 3.3, cea încărcată de utilizator din galeria telefonului.

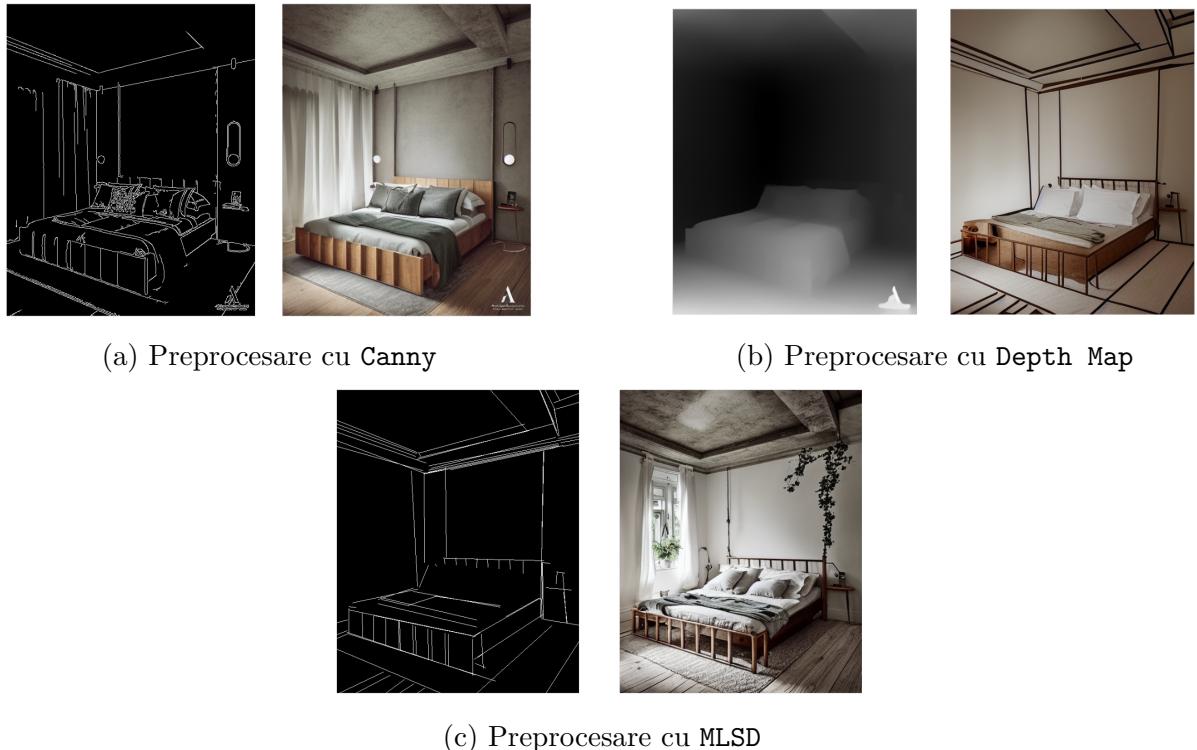


Figura 3.5: Comparație între metodele de condiționare a imaginii inițiale în ControlNet

Metoda Canny oferă un contur clar al obiectelor, dar se dovedește sensibilă la detaliu inutile din fundal, ceea ce poate afecta coerența rezultatului. Depth Map introduce o reprezentare bazată pe adâncime, utilă pentru înțelegerea spațiului, însă lipsită de precizia geometrică necesară în amenajări interioare. În schimb, MLSD oferă o mapare structurală fidelă a liniilor drepte predominante (pereți, colțuri, mobilier), păstrând cu acuratețe arhitectura camerei originale.

Astfel, în cadrul aplicației *Interio*, a fost aleasă varianta MLSD ca metodă de condiționare principală, întrucât oferă cele mai bune rezultate în menținerea structurii spațiului, esențiale pentru generarea unor imagini realiste și funcționale.

În plus, în imaginile generate cu procesare MLSD se observă frecvent apariția unor obiecte suplimentare precum plante sau elemente decorative, chiar dacă acestea nu apar în imaginea originală. Acest comportament se explică prin faptul că MLSD oferă un ghid geometric clar, care permite modelului AI să plaseze obiecte noi într-un mod realist, conform descrierii din prompt.

În plus, pentru a obține rezultate vizuale mai realiste și personalizate, a fost adusă o îmbunătățire semnificativă în procesul de construire a prompturilor trimise către modelul AI, prin aplicarea unor tehnici de prompt engineering. În locul unui prompt generic, aplicația *Interio* construiește automat un mesaj detaliat care include: tipul camerei, stilul de design selectat, o descriere specifică stilului ales, denumirea paletelor de culori preferate și eventuale indicații suplimentare introduse de utilizator. Fiecare stil are asociat un fragment text unic care descrie atmosfera și detaliile vizuale ale stilului respectiv.

## 4. Concluzii

Aplicația Interio oferă o soluție inovatoare pentru utilizatorii care doresc să își amenajeze locuința într-un mod personalizat și eficient, bazându-se pe inteligență artificială și preferințele stilistice ale acestora. Îmbinând tehnici moderne de clasificare și generare de imagini, aplicația reușește să simplifice un proces adesea complex, oferind o experiență intuitivă și interactivă pentru utilizator.

Din punct de vedere al accesibilității, Interio este ușor de utilizat, fiind dezvoltată ca aplicație mobilă *cross-platform*, compatibilă cu dispozitive Android și iOS. Interfața este prietenoasă, minimalistă și intuitivă, gândită pentru a ghida utilizatorul pas cu pas în procesul de personalizare și generare a unei amenajări. Fluxul de utilizare este logic și clar structurat, astfel încât atât utilizatorii tehnici, cât și cei fără experiență digitală avansată să poată interacționa cu aplicația fără dificultăți.

În ceea ce privește securitatea, mecanismul de autentificare bazat pe JSON Web Tokens (JWT) asigură protejarea informațiilor personale, iar arhitectura client-server *stateless* contribuie la un sistem scalabil și rapid. Validarea inputului utilizatorului este prezentă în toate formularele aplicației, prevenind atacurile de tip injection sau introducerea de date invalide. Datele transmise între client și server sunt protejate prin HTTPS, ceea ce asigură confidențialitatea comunicației. În plus, accesul la date sensibile este controlat prin politici de autorizare bine definite, iar parolele utilizatorilor sunt stocate în mod securizat folosind hashing (prin mecanismele oferite de ASP.NET Identity).

Pentru a asigura o experiență de utilizare coerentă și fără erori, aplicația *Interio* a fost supusă unui proces de testare manuală pe multiple dispozitive. Fiecare funcționalitate principală a fost testată în scenarii reale de utilizare, pentru a identifica eventuale bug-uri sau incoerențe în interfață. Testarea a vizat atât comportamentul aplicației în condiții normale de utilizare, cât și tratarea corectă a erorilor (ex: câmpuri necomplete, parole incorecte, imagini lipsă etc.). Observațiile obținute în urma acestei etape au contribuit semnificativ la creșterea calității aplicației.

În ceea ce privește îmbunătățirile viitoare, o primă propunere vizează introducerea unor noi stiluri de design interior, mai variate, pentru a acoperi și mai bine diversitatea preferințelor utilizatorilor. De asemenea, o funcționalitate utilă reprezintă afișarea detaliilor despre piesele de mobilier prezente în imaginile generate, împreună cu linkuri către magazine online de unde acestea pot fi achiziționate. Această integrare poate fi realizată prin conectarea la API-uri oferite de companii de mobilier, ceea ce deschide oportunități de monetizare parteneriatelor cu magazinele online ale acestor companii. În acest mod, aplicația *Interio* ar evoluă de la un simplu generator vizual la un instrument complet de amenajare, care le permite utilizatorilor să își transforme ideile în realitate, direct din aplicație.

# Bibliografie

- [1] *ASP.NET Core Documentation*, Accesată la data [2 iunie 2025], URL: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0>.
- [2] *ASP.NET Identity Documentation*, Accesată la data [2 iunie 2025], URL: <https://learn.microsoft.com/en-us/aspnet/identity/>.
- [3] *Flask Documentation*, Accesată la data [2 iunie 2025], URL: <https://flask.palletsprojects.com/en/stable/>.
- [4] Ian J. Goodfellow et al., *Generative Adversarial Networks*, Accesată la data [2 iunie 2025], 2014, URL: <https://arxiv.org/abs/1407.7502>.
- [5] Michael B. Jones, John Bradley și Nat Sakimura, *JSON Web Token (JWT)*, <https://datatracker.ietf.org/doc/html/rfc7519>, Accesată la data [2 iunie 2025], 2015.
- [6] Zichun Shao Mengchao Ruan Huiting Li Dong Zheng Yanyan Liang Junming Chen Xiaodong Zheng, *Creative interior design matching the indoor structure generated through diffusion model with an improved control network*, Accesată la data [8 iunie 2025], 2024, URL: <https://www.sciencedirect.com/science/article/pii/S2095263524001171>.
- [7] Vadim Sokolov Maria Nareklishvili Nick Polson, *Generative Modeling: A Review*, Accesată la data [7 iunie 2025], 2025, URL: <https://arxiv.org/abs/2501.05458>.
- [8] Microsoft, *Entity Framework Core - Migrations Overview*, Accesată la data [2 iunie 2025], 2024, URL: <https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/>.
- [9] *Microsoft SQL Server Documentation*, Accesată la data [2 iunie 2025], URL: <https://learn.microsoft.com/en-us/sql/sql-server/>.
- [10] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- [11] S. Rasoul Safavian și David R. Landgrebe, *A survey of decision tree classifier methodology*, Accesată la data [2 iunie 2025], 2005, URL: [https://www.researchgate.net/publication/225237661\\_Decision\\_Trees](https://www.researchgate.net/publication/225237661_Decision_Trees).
- [12] Jurgen Schmidhuber, *Deep Learning in Neural Networks: An Overview*, Accesată la data [7 iunie 2025], 2014, URL: <https://arxiv.org/abs/1404.7828>.
- [13] *UI Kitten Documentation*, Accesată la data [2 iunie 2025], URL: <https://akveo.github.io/react-native-ui-kitten/docs/getting-started/what-is-ui-kitten#what-is-ui-kitten>.

- [14] Helen Ying, *Random Forest Algorithm: A Complete Guide*, Accesată la data [26 mai 2025], 2023, URL: <https://builtin.com/data-science/random-forest-algorithm>.
- [15] Lvmin Zhang și Maneesh Agrawala, *Adding Conditional Control to Text-to-Image Diffusion Models*, Accesată la data [3 iunie 2025], 2023, URL: <https://arxiv.org/abs/2302.05543>.

# Anexe

## Anexa 1 - Interfața aplicației

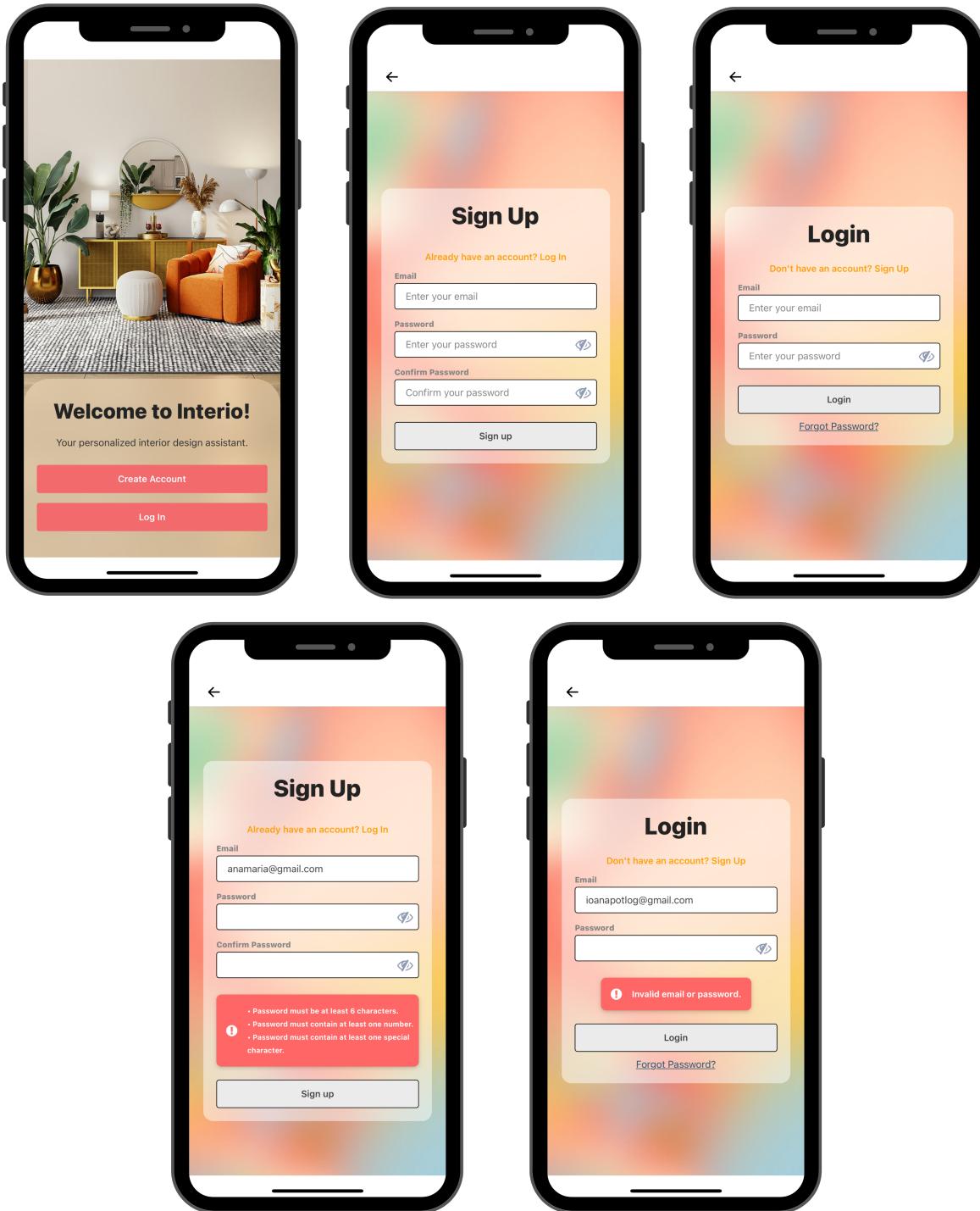


Figura 1. Fluxul UI al funcționalității de autentificare

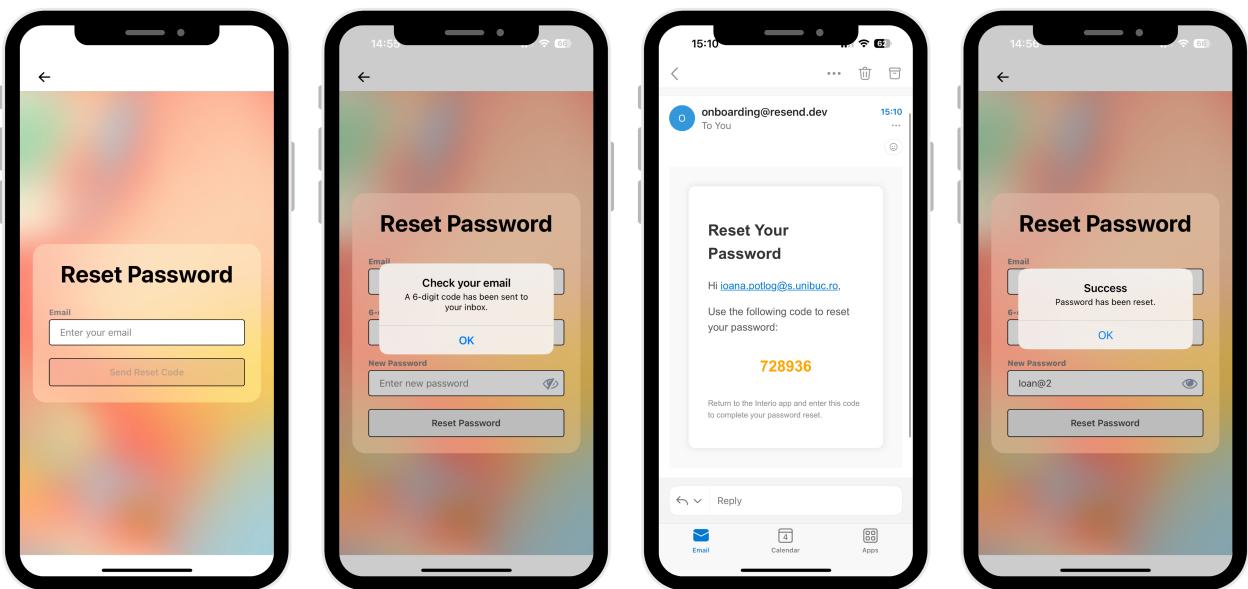


Figura 2. Fluxul UI al funcționalității de resetare a parolei

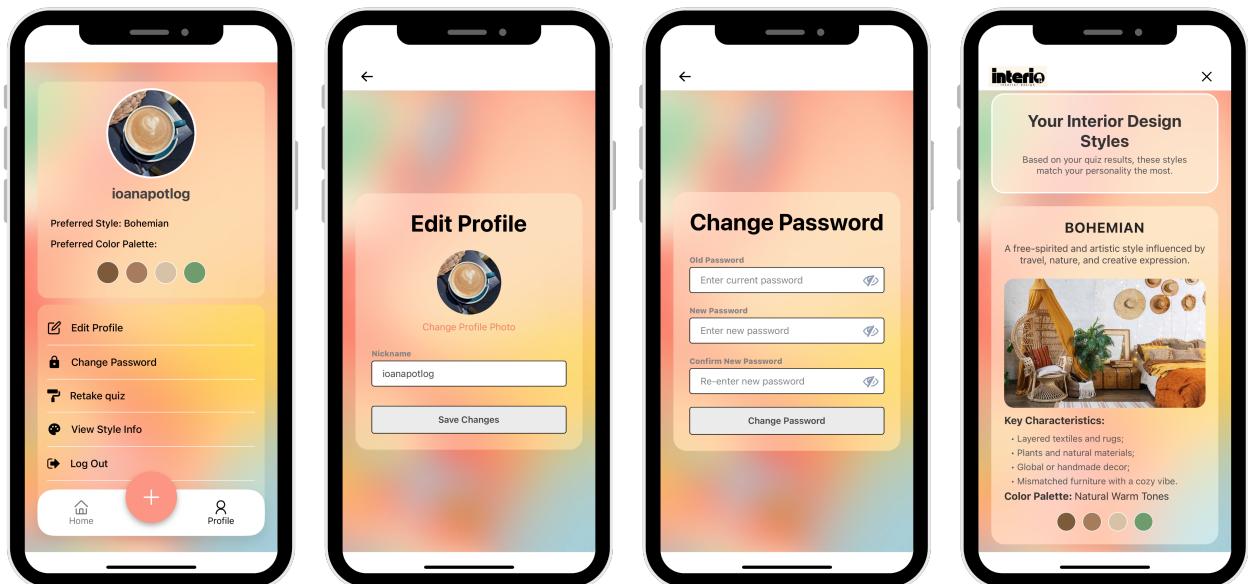


Figura 3. Fluxul UI al funcționalității de gestionarea a profilului utilizatorului

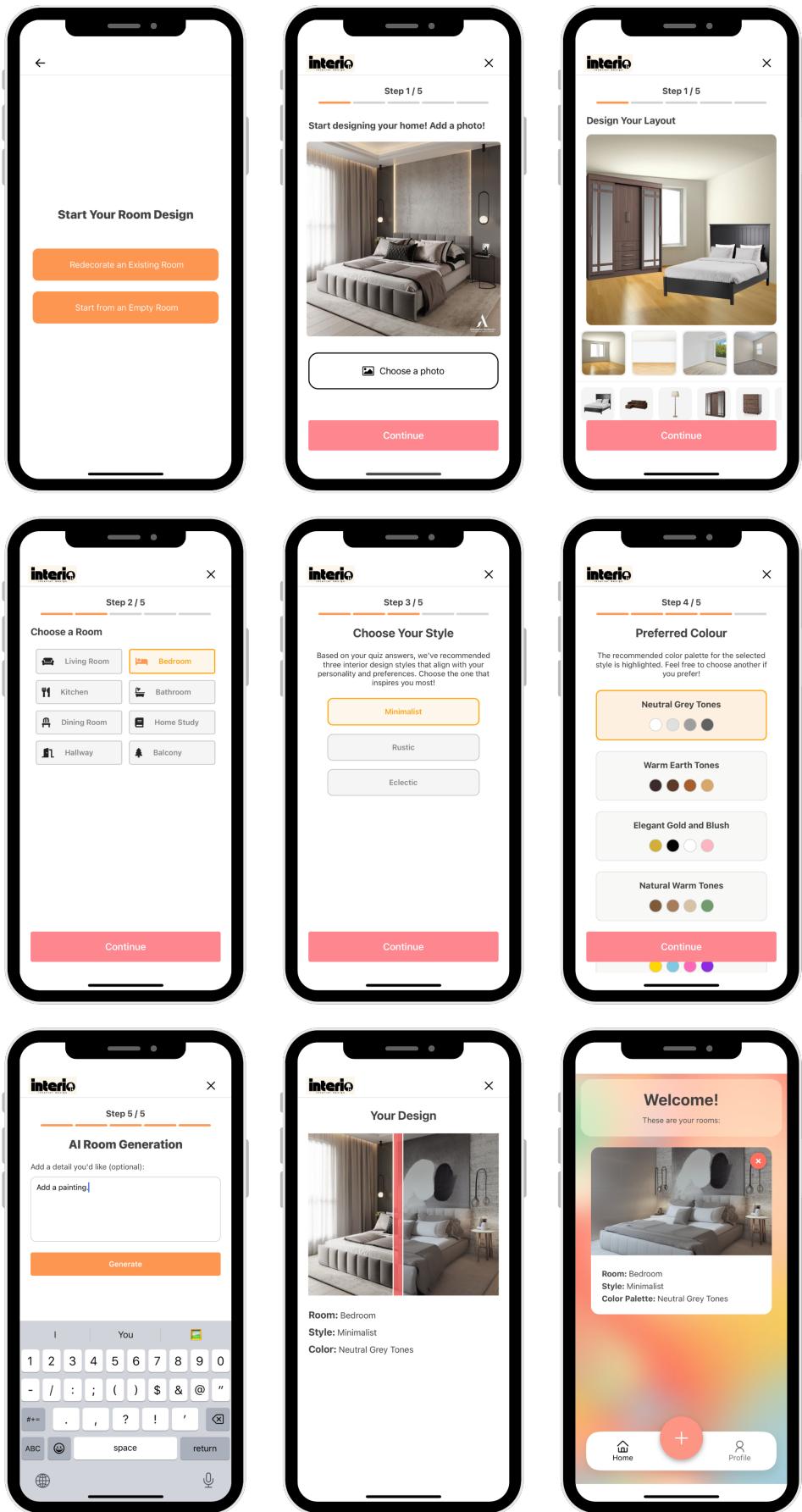


Figura 4. Fluxul UI al funcționalității de generare a imaginii cu camera redecorată

## Anexa 2 - Secvențe de cod

```
1 import numpy as np
2 import pandas as pd
3 from sklearn.preprocessing import LabelEncoder
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.model_selection import train_test_split
6
7 # Load data
8 df = pd.read_excel("QuizStyleTrainingData.xlsx")
9
10 x = df[[f"Q{i}" for i in range(1, 9)]]
11 y = df["Style"]
12
13 label_encoder = LabelEncoder()
14 y_encoded = label_encoder.fit_transform(y)
15
16 # Train the model
17 clf = RandomForestClassifier(n_estimators=100, random_state=42)
18 clf.fit(x, y_encoded)
19
20 def predict_styles(answers: list[int]):
21     input_df = pd.DataFrame([answers], columns=[f"Q{i}" for i in range(1, 9)])
22
23     probs = clf.predict_proba(input_df)[0]
24
25     top3_indices = np.argsort(probs)[-3:]
26     top3_styles = label_encoder.inverse_transform(top3_indices)
27     print("Top 3 predicted styles:", top3_styles)
28
29     return top3_styles.tolist()
```

Figura 5. Secvență de cod — clasificator Random Forest

```
1 import torch
2 from controlnet_aux import MLSDdetector
3 from diffusers import ControlNetModel, StableDiffusionControlNetPipeline, UniPCMultistepScheduler
4 from PIL import Image
5
6 def upscale_image(image, scale_factor=2):
7     width, height = image.size
8     new_size = (width * scale_factor, height * scale_factor)
9     upscaled_image = image.resize(new_size, Image.LANCZOS)
10    return upscaled_image
11
12 def generate_furnished_room(image, prompt, output_buffer):
13     print(prompt)
14
15     mlsd = MLSDdetector.from_pretrained('llyasviel/ControlNet')
16     control_image = mlsd(image)
17
18     controlnet = ControlNetModel.from_pretrained("llyasviel/control_v1p_sd15_mlsd", torch_dtype=torch.float16)
19     pipe = StableDiffusionControlNetPipeline.from_pretrained(
20         "runwayml/stable-diffusion-v1-5", controlnet=controlnet, torch_dtype=torch.float16
21     )
22
23     pipe.scheduler = UniPCMultistepScheduler.from_config(pipe.scheduler.config)
24     pipe.enable_model_cpu_offload()
25
26     generator = torch.manual_seed(0)
27     result = pipe(prompt, num_inference_steps=50, guidance_scale=8.5, generator=generator, image=control_image).images[0]
28
29     upscaled_result = upscale_image(result)
30     upscaled_result.save(output_buffer, format="PNG")
```

Figura 6. Secvență de cod — generarea imaginilor cu Stable Diffusion și ControlNet

## Anexa 3 - Structura bazei de date

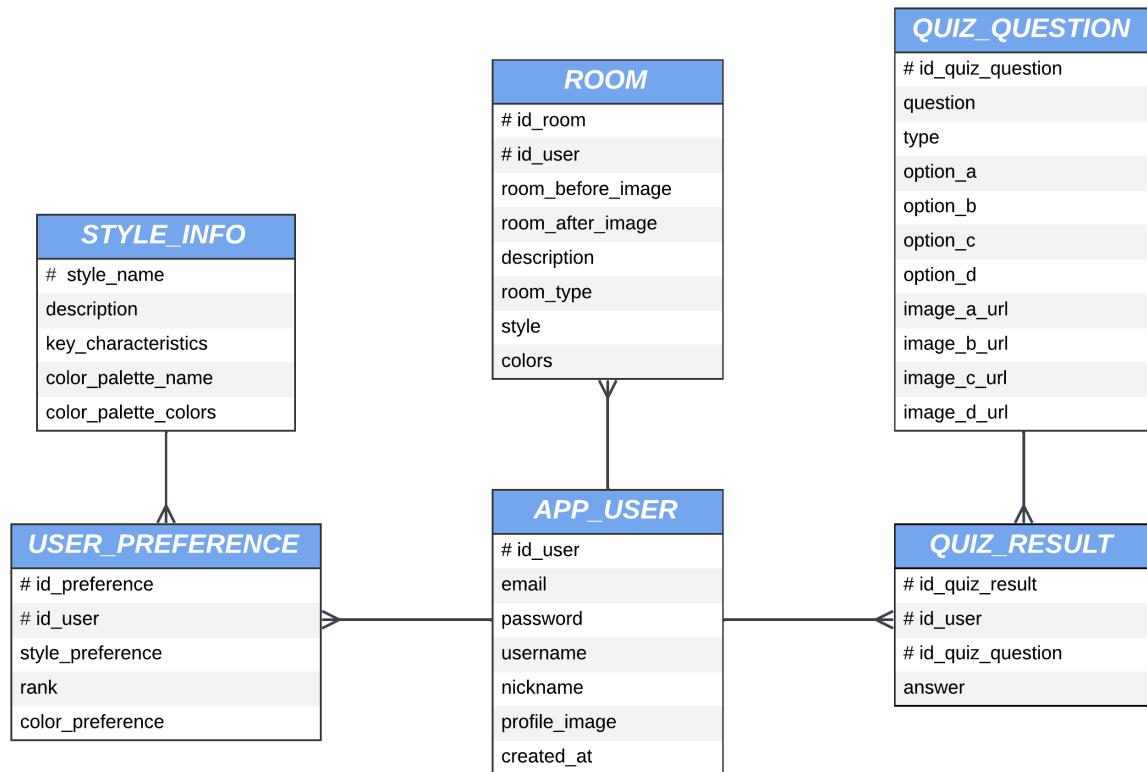


Figura 7. Diagrama entitate-relație (ERD) — structura bazei de date