

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA



LUCRARE DE LICENTA

Platformă interactivă de desen digital:

BlankCanva

propusă de

Ana-Maria Ghiorghiu

Sesiunea: iunie/iulie, 2023

Coordonator științific

Lect. dr. Cosmin Vârlan

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA

**Platformă interactivă de desen digital:
BlankCanva**

Ana-Maria Ghiorghiu

Sesiunea: iunie/iulie, 2023

Coordonator științific

Lect. dr. Cosmin Vârlan

Cuprins

1 Introducere	2
1.1 Motivatie	2
1.2 Contribuții și abordări similare	3
1.2.1 Contextul de blog	3
1.2.2 Contextul de desenat	4
1.3 Obiective generale	5
1.3.1 Cerințe funcționale	6
1.4 Abordarea tehnică a aplicației	7
1.5 Structura lucrării	8
2 Aspecte tehnice ale tehnologiilor utilizate	9
2.1 TypeScript	9
2.2 Next.js	11
2.2.1 React	11
2.2.2 Next.js vs React	12
2.3 Tailwind CSS	14
2.3.1 Comparație din punct de vedere aspectual	14
2.3.2 Comparație din punct de vedere al rației dintre cod și stilizare	15
2.4 Librării externe	15
2.4.1 dayjs	15
2.4.2 slugify	16
2.4.3 react-hot-toast	17
2.5 Concluzie	17
3 Dezvoltarea aplicației web	19
3.1 Arhitectura Server-Client	19
3.2 Modelul C4 pentru vizualizarea arhitecturii	21

3.3	Detalii tehnice ale arhitecturii	22
3.3.1	Autentificarea în aplicație	23
3.3.2	Arhitectura bazei de date	26
3.4	Virtualizarea aplicației la nivel de sistem de operare	29
3.5	Concluzie	30
4	Componenta de desen digital	32
4.1	Proiectarea aplicației de desenat	32
4.2	Implementarea aplicației de desenat	35
4.2.1	Planșa de desen	35
4.2.2	Salvarea planșei de desen	37
4.3	Concluzie	38
5	Inteligenta artificială în arta digitală	40
5.1	Rețele Generative Adversariale	40
5.2	Difuziune	42
5.2.1	Comparație între GAN și difuziune	43
5.3	Stable Diffusion	44
5.3.1	Prompt	44
5.3.2	Negative Prompt	45
5.3.3	Modelul ales	46
5.4	ControlNet	47
5.4.1	Preprocesorul și Modelul ales	48
5.5	Concluzie	48
6	Direcții de viitor	49
Concluzii		51
Bibliografie		52

Capitolul 1

Introducere

În era digitală contemporană, arta a progresat în moduri fără precedent, preluând multiple forme și având o continuă evoluție. Una dintre cele mai reprezentative forme de exprimare este pictura digitală, care a dobândit o importanță deosebită de semnificativă în societate.

În acest context, lucrarea mea de licență propune să creeze un pod între creatori și admiratori de artă, printr-o platformă dedicată lor.

Când am făcut pentru prima dată o pictură digitală în cadrul facultății, am considerat că folosirea inteligenței artificiale în crearea acestor desene este intimidantă, dată fiind gama vastă de tehnici complexe.

BlankCanva intenționează să ajute prin dezvoltarea unui blog pentru aceste picturi folosind inteligență artificială ce ajuta în procesul de creare artistică, împreună cu posibilitatea de a împărtăși creațiile personale cu restul utilizatorilor, dar și opțiunea de a oferi *feedback* pentru lucrările postate de acestia.

"Noi nu suntem conștienți de modul în care desfășurăm anumite activități, cum ar fi înțelegerea limbajului, recunoasterea *pattern*-urilor și aşa mai departe, dar avem tehnici de inteligență artificială tot mai bune capabile să reproducă astfel de activități."

1.1 Motivație

Blogul alcătuit exclusiv din postări cu desene digitale ca și lucrare de licență are ca motivație o combinație de pasiuni personale. Am fost mereu atrasă de intersecția dintre tehnologie și artă, și a fost de interes pentru mine modul în care noile tehnologii

pot ajuta la dezvoltarea artei. Desenul digital, în special, mi s-a părut o formă de artă care combină abilitățile artistice tradiționale cu abilitățile tehnice.

O altă motivație importantă în alegerea mea, se bazează pe convingerea că talentul artistic nu ar trebui să fie o barieră pentru exprimarea creativă și participarea în comunități de artă digitală. De aceea, aplicația pe care o dezvolt în cadrul acestei lucrări de licență integrează inteligența artificială pentru a ajuta utilizatorii să aibă cele mai reușite lucrări.

Inteligenta artificială va avea rolul de a oferi o lucrare finalizată bazată pe desenul utilizatorului. Astfel, rolul utilizatorului este redefinit: el nu mai este doar un creator, ci și un administrator, alegând sugestia generată care se potrivește cel mai bine cu viziunea sa artistică.

BlankCanva

Figura 1.1: Sigla aplicației, BlankCanva

Titlul aplicației mele, *BlankCanva* face referire la felul în care orice utilizator își poate picta vizunile pe pânza infinită a monitorului său. Pânza albă este punctul de plecare în crearea oricărei opere de artă, însă în acest context, aceasta așteaptă artistul înarmat cu pensule virtuale și palete de culori digitale să își afișeze ideile.

Consider că această lucrare nu numai că va permite să aplic cunoștințele mele tehnice și creative, dar va și contribui la dezvoltarea comunității artistice.

1.2 Contribuții și abordări similare

1.2.1 Contextul de blog

Există în prezent multe platforme similare, pe care artiștii își împărtășesc arta. Printre cele mai utilizate aplicații de acest fel în industrie, la momentul actual, este *DeviantArt*.

Fondată în anul 2000, *DeviantArt* este printre cele mai mari comunități online pentru artiști, având milioane de utilizatori și postări încărcate, prin intermediul căreia utilizatorii pot să își dezvăluie și să discute creațiile lor artistice.

Voi prezenta în tabelul ce urmează asemănările și deosebirile principale dintre site-ul menționat și *BlankCanva*.

Asemănări	Deosebiri
<ul style="list-style-type: none"> Cele două platforme permit schimbul de idei, comentarii, <i>feedback</i> pe pagina principală a unei postări și salvarea postărilor de interes într-o secțiune personală de <i>bookmarks</i>. Cele două platforme permit artiștilor să creeze un profil online cu lucrările personale. Cele două platforme permit filtrarea postărilor după cuvinte cheie și alcătuirea unei liste de prieteni, fiind artiștii care au atras atenția prin lucrările deosebite, și se dorește urmărirea viitoarelor postări din partea lor. 	<ul style="list-style-type: none"> <i>BlankCanva</i> adoptă politica diferită de a putea posta doar lucrările create în cadrul aplicației, astfel toate postările sunt din același domeniu, păstrând <i>BlankCanva</i> ca un spațiu virtual dedicat picturii digitale în totalitate. <i>BlankCanva</i> pune la dispoziție o aplicație de <i>Paint</i>, în cadrul căreia utilizatorul poate desena. <i>BlankCanva</i> pune la dispoziție funcționalitatea de aplicare a inteligenței artificiale asupra desenului creat, având ca rezultat pictura îmbunătățită.

Tabela 1.1: Asemănări și deosebiri între *DeviantArt* și *BlankCanva*

1.2.2 Contextul de desenat

Probabil cea mai similară aplicație pentru desenat este *Microsoft Paint*. Am dorit ca această componentă din proiect să fie o variantă simplificată și incorporată a aplicației pe care utilizatorii de *Windows* o dețin în calculator.

Microsoft Paint a fost inclus în *Windows* de câteva decenii, această aplicație oferă utilizatorilor posibilitatea de a crea și edita grafică.

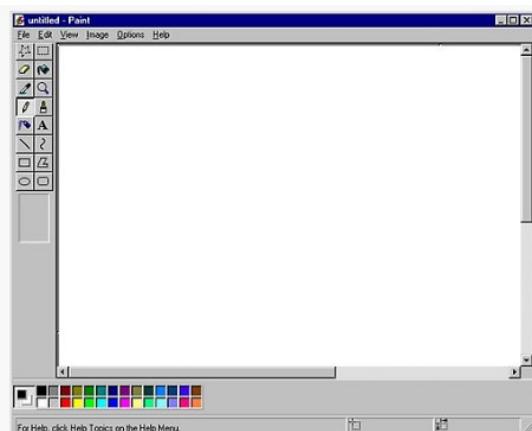


Figura 1.2: *Microsoft Paint*

Paint este simplu de utilizat, are toate instrumentele *default* și reprezintă o opțiune atractivă pentru persoanele care sunt la început în ceea ce privește editarea.

Paint-ul meu, în contrast cu *Microsoft Paint*, dispune de opțiunea de straturi (*layers*).

Acest *tool* adăugă un nivel suplimentar de flexibilitate în proiectarea desenului și permite organizarea elementelor grafice în straturi separate și editarea sau ștergerea individuală.

De asemenea, o îmbunătățire pentru *Microsoft Paint*, este opțiunea de a salva o fotografie într-o bază de date și să le poți vizualiza într-o galerie, fără a fi nevoie să descarci de fiecare dată produsul final.

Menirea proiectului nu este de a lucra în *Paint* exclusiv, ci de a observa diferențele majore de după modificările făcute de componenta de inteligență artificială. Astfel componenta de desen din cadrul proiectului, vine cu funcționalitățile principale create pentru ca aceasta să devină o unealtă de a exprima ideea pe care o dorim într-o imagine creată de *Stable Diffusion*.

1.3 Obiective generale

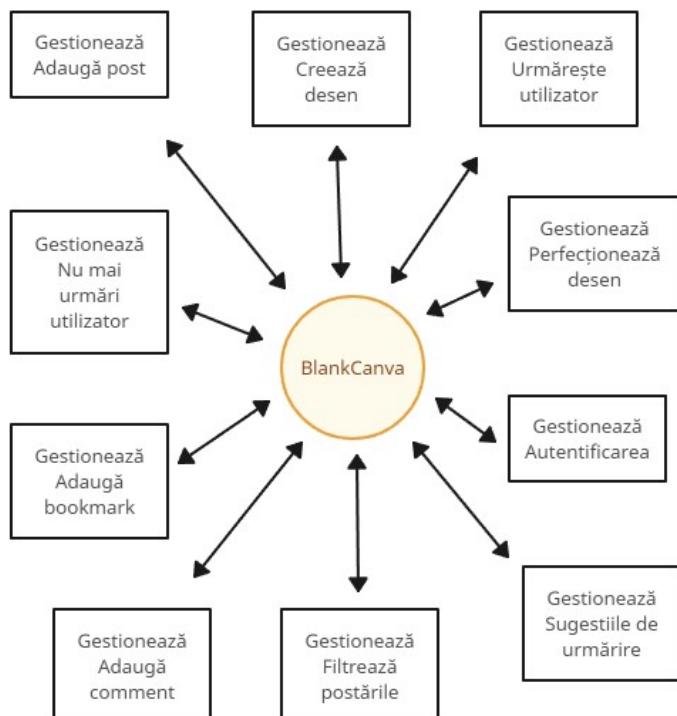


Figura 1.3: Principalele obiective ale aplicației

Cele trei cuvinte cheie care definesc acest proiect sunt *desen*, *îmbunătățire* și *partajare*. Pentru a susține această idee, obiectivele generale ale acestui proiect sunt:

- Dezvoltarea unei platforme intuitive
- Înțelegerea procesului din spate a tehnologiilor de artă generativă
- Implementarea unui sistem de îmbunătățire a desenelor cu ajutorul inteligenței artificiale
- Crearea unui spațiu de partajare a desenelor
- Promovarea unei comunități active
- Asigurarea securității datelor utilizatorilor ce se autentifică prin *Google*

1.3.1 Cerințe funcționale

Autentificarea în aplicația *BlankCanva* este un proces facil datorită integrării serviciilor *Google*. Un utilizator nou, conectat prin contul de *Gmail*, este întâmpinat de comunitatea de creatori de artă digitală.

Pagina principală este un flux viu de postări ale utilizatorilor, care afișează noutățile de fiecare dată când se adaugă o nouă postare. În momentul în care un utilizator adaugă în lista sa de *bookmarks* o postare deosebită, *BlankCanva* generează sugestii personalizate de creatori cărora să le dea *follow*.

Platforma include un modul de creare a desenelor digitale, similar cu un program de tip *Paint*. Atunci când creatorul consideră că pictura este finalizată, va putea alege varianta perfectă retușată de *AI* a desenului său.

Aplicația oferă fiecărui utilizator o pagină personală, o pagină unde poate vizualiza toate desenele pe care le-a creat, iar pe pagina unui alt utilizator va putea vedea postările făcute de acesta.

Pentru a facilita descoperirea de postări pe gustul utilizatorului, platforma permite filtrarea postărilor după anumite etichete (*tags*). În același timp, există opțiunea de a anula filtrele și de a vedea toate postările. Atunci când un utilizator face o postare, poate adăuga propriul *tag*, existent sau nou creat, pentru a ajuta alți utilizatori interesați să găsească postarea sa.

1.4 Abordarea tehnică a aplicației



BlankCanva este o aplicație web dezvoltată cu ajutorul **T3 Stack**, un set de dezvoltare ce vizează aplicațiile *full-stack*. Pentru partea de frontend am ales TypeScript și Next.js, pentru stilizarea aplicației am ales Tailwind CSS, iar pentru partea de backend, baze de date și autentificare am ales tRPC, Prisma și NextAuth.js.



Next.js este un *framework* pentru construirea aplicațiilor web, ce permite construirea interfețe de utilizator folosind componente *React*. *Next.js* oferă optimizări și configuraază automat instrumentele precum compilarea.



TypeScript este un limbaj de programare care se bazează pe *JavaScript*, diferența principală fiind faptul că adaugă o sintaxă suplimentară, fiind necesare tipurile de date, cu scopul de a înțelege, observa și recunoaște eventualele erori.



Tailwind CSS funcționează prin scanarea tuturor fișierelor ce conțin structuri *HTML*, componentelor *JavaScript/TypeScript* și oricărora alte şabloane pentru nume de clase, generând stilizarea corespunzătoare și apoi scriindu-le într-un fișier *CSS static*.



tRPC, sau *TypeScript Remote Procedure Call* este cea mai simplă bibliotecă pentru apelarea la distanță a funcțiilor backend pe partea de client, făcând comunicarea între *backend* și *frontend* mai rapidă și efectivă.



Prisma permite dezvoltatorilor să-și definească modelele de aplicații într-un limbaj intuitiv de modelare a datelor.



NextAuth.js este o soluție de autentificare completă *open-source* pentru aplicațiile *Next.js*.

1.5 Structura lucrării

Lucrarea este structurată în 6 capitole, iar primul capitol este introducerea în lucrarea mea de licență, cele ce urmează fiind descrise sumar astfel:

În al doilea capitol al lucrării, intitulat *Aspecte tehnice ale tehnologiilor utilizate* voi aprofunda aspectele tehnice ale lucrării, și mă voi axa pe avantajele acestora. Primul proiect în care s-a folosit T3 Stack, datează din vara anului 2022. Prin urmare, voi prezenta ce caracteristici m-au determinat să aleg acest pachet, și ce librării externe am folosit în cadrul aplicației.

În al treilea capitol al lucrării, intitulat *Dezvoltarea aplicației web*, voi explora mai amănuntit conceptele legate de aplicație. Voi detalia tehnologiile incluse în procesul de autentificare, iar mai departe în acest capitol, voi prezenta și baza de date, de la alegerile făcute pentru baza de date până la aspectul arhitecturii proiectului.

În primul subcapitol *Arhitectura Server-Client*, este arătat un exemplu de comunicare între server și client pe care l-am urmat pe tot parcursul proiectului. În cel de-al doilea subcapitol, *Modelul C4 pentru vizualizarea arhitecturii*, se găsesc diagramele C4 ale proiectului, iar în următorul subcapitol, *Arhitectura generală* vom găsi o diagramă *use-case* care ilustrează sugestiv funcționalitățile aplicației și cum interacționează pentru user, procesul de autentificare amănuntit, și arhitectura bazei de date. În ultimul subcapitol *Virtualizarea aplicației la nivel de sistem de operare*, am parcurs procesul de virtualizare a unui proiect prin tehnologia de containerizare.

În al patrulea capitol al lucrării, intitulat *Componenta de desen digital*, voi detalia componenta de desen din aplicație, adică tehnologiile folosite pentru crearea ei, precum și o diagramă care descrie funcționalitățile acestei componente și felul în care aceasta funcționează. Împărțit în două subcapitole, *Proiectarea aplicației de desenat* și *Implementarea aplicației de desenat*, primul subcapitol vizează arhitectura și funcționalitățile, fiecare instrument în parte și rolul lui, iar al doilea subcapitol vizează tehnologiile folosite în desen și tehnica utilizată pentru salvarea planșei.

În al cincilea capitol, *Inteligenta artificială în arta digitală*, vom cuprinde toate aspectele teoretice și practice legate de componenta de inteligență artificială a proiectului. De asemenea, voi prezenta succint procesul de creare al imaginilor și ce funcționalități am folosit din Stable Diffusion, și rolul extensiei ControlNet.

În al șaselea capitol voi menționa ce direcții de viitor aş vrea să aibă aplicația și unde consider că se pot aduce îmbunătățiri.

Capitolul 2

Aspecte tehnice ale tehnologiilor utilizate

Research-ul legat de funcționalitățile proiectului a determinat să mă orientez către un pachet de dezvoltare web care să aducă beneficii pentru o platformă de *sharing* între utilizatori. Proiectul este de tip full-stack, iar T3 stack oferă o structură ușor de urmărit și o comunicare foarte eficientă între Frontend și Backend.

T3 Stack a fost menit să livreze un punct de pornire solid în dezvoltarea unei aplicații prin unirea celor 6 tehnologii în cadrul aceleiași aplicații. Însă, este alegerea fiecărui utilizator dacă dorește să utilizeze toate pachetele, iar alegerea o poate face chiar în momentul în care acesta își crează aplicația.

Pentru a crea aplicația, este suficientă o simplă comandă:

```
npm create t3-app@latest
```

2.1 TypeScript

T3 Stack are ca bază limbajul de programare TypeScript. TypeScript aduce numeroase avantaje ca și limbaj de programare al Frontend-ului. Fiind intuitiv, este o alegere bună în contextul creării de pagini dinamice și interactive.

Unul dintre aceste avantaje este că în eventualele erori care pot apărea, vom ști precis și în manieră instantă că este posibil să fie o problemă de la tipizarea gresită sau pasarea unui argument de un tip greșit într-o funcție. Poate fi considerat o adiție către JavaScript, practic se bazează pe aceleasi principii, aducând o sintaxă adițională. În sine, TypeScript este compilat în JavaScript înainte de rulare, însă există un beneficiu

major în a-l folosi în favoarea JavaScript-ului. Într-o situație în care se rulează o funcție JavaScript cu datele pasate de tipuri greșite, această funcție va furniza un output, fără a ridica o eroare specifică precum *TypeError*.

În secvența de mai jos, arăt începutul unei componente aşa cum e implementat în proiectul meu, și cum ar fi arătat acesta dacă era scris în JavaScript:

```
1 interface ComponentProps {  
2   base64Image: string;  
3 }  
4  
5 const Component: React.FC<ComponentProps> = ({ base64Image }) => {  
6   const [prompt, setPrompt] = useState<string | null>(null);  
7   const [response, setResponse] = useState<string | null>(null);  
8   const [loading, setLoading] = useState<boolean>(false);  
9  
10  .  
11  .  
12  }
```

Listing 2.1: Componentă TypeScript

```
1 const YourComponent = ({ base64Image }) => {  
2   const [prompt, setPrompt] = React.useState(null);  
3   const [response, setResponse] = React.useState(null);  
4   const [loading, setLoading] = React.useState(false);  
5  
6   .  
7 }
```

Listing 2.2: Componentă JavaScript

Se observă din secvențele anterioare diferența dintre cele două și cum pot afecta mai departe parcursul implementărilor.

Un alt avantaj este că orice cod JavaScript este de asemenea cod TypeScript valid. TypeScript este un superset al JavaScript, ceea ce va însemna faptul că acesta include toate funcționalitățile JavaScript și adaugă alte caracteristici în plus, cum ar fi tipurile statice și interfețele.

Tabelul de mai jos ilustrează și alte diferențe dintre JavaScript și TypeScript:

Tabela 2.1: Diferențe între JavaScript și TypeScript

Proprietăți	JavaScript	TypeScript
Suport la parametri	Nu	Da
Gestionarea erorilor	Se evidențiază după compilare	Se evidențiază instant
OOP	Prototip bazat pe OOP	OOP clasic
Siguranță la tipare	Mai redusă	Mai sporită
Suport pentru generics	Nu	Da
Module	Suport limitat	Suport extins

2.2 Next.js

Despre *Next.js* s-ar putea spune la prima vedere că este *React*, dar *Next.js* este construit **folosind React**, ceea ce face ca cele două să aibă foarte multe asemănări. *Next.js* este un framework *React* pentru construirea aplicațiilor web, și deci, toate caracteristicile *React* vor fi disponibile în acesta. În acest subcapitol vom observa care sunt proprietățile pe care le aduce în plus *Next.js*.

2.2.1 React

React este o bibliotecă *open-source* dezvoltată de *Facebook*, special menită pentru construirea interfețelor de utilizator. *React* este cel mai popular *framework* în acest domeniu în prezent și este folosit pentru a crea aplicații web cu o singură pagină. Prin *single-page* ne referim la faptul că aplicația va avea un singur document HTML și îl actualizează după cum este necesar.

React îmbunătăște performanța unei aplicații datorită *virtual DOM (Document Object Model)*, care este cu siguranță mult mai rapid decât *HTML DOM*.

HTML DOM poate fi reprezentat ca un arbore de noduri, iar fiecare nod reprezintă câte o componentă (poate fi orice componentă html, `<div>`, `<section>`, `<p>`, `<h1>`, `<head>`, `<title>`, etc).

```

<html>
  <head>
    <title></title><link></link>
  </head>
  <body>
    <n>
      <div><a><img></a></div>
    </figure>
    <table>
      <tbody>
        <tr><td></td><td></td><td></td></tr>
        <tr><td></td><td></td><td></td></tr>
      </tbody>
    </table>
  </body>
</html>

```

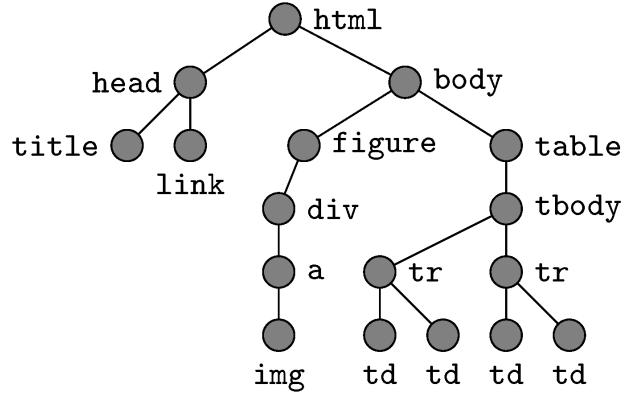


Figura 2.1: Exemplu DOM (preluat)

DOM-ul virtual stocă o reprezentare a interfeței utilizator în memorie și este sincronizat cu *DOM*-ul actual cu ajutorul *React DOM*.

HTML DOM are ca mecanism de funcționare modificarea unuia dintre noduri, care va genera inițial o parcurgere a arborelui, și va căuta nodurile ce trebuie modificate, după care va fi necesară adăugarea de noi elemente, atribute și a conținutului, iar abia după, nodurile *DOM*-ului vor putea fi actualizate.

Atunci când arborele unei pagini încărcate de componente, devine foarte larg, această tehnică este costisitoare din punct de vedere al timpului, resurselor și al memoriei.

Într-o aplicație contemporană, este necesară actualizarea constantă a aplicației, și optimizarea constă în lucrul direct cu memoria, aşa cum face *Virtual DOM*. *DOM*-ul virtual este o copie a *DOM*-ului *HTML*.

Când există orice actualizări, *React* le actualizează mai întâi în *DOM*-ul virtual și le compară cu *DOM*-ul real, apoi *React* va actualiza *DOM*-ul real numai dacă există modificări și doar părțile modificate. Astfel, efectuează mai puține operații pe *DOM*-ul real, reducând semnificativ timpul de actualizare.

2.2.2 Next.js vs React

Pentru a ilustra mai ușor diferențele și asemănările, voi pune cele principale într-un tabel, precum și adăugirile care le demonstrează:

Proprietate	React	Next.js
Cod*	Este relativ ușor de implementat aplicații cu React dacă se cunosc particularitățile din JavaScript	Cum Next.js este construit pe React, codul și structura este asemănătoare
Structura folderelor	Nu sunt impuse reguli cu privire la structura proiectului	Impune o structură specifică care trebuie urmărită, datorită procesului de <i>Pages Routing</i>
TypeScript	Sustine TypeScript și JavaScript	Sustine doar TypeScript
Performanță	Ușor mai lent	Mai avansat decât React
Mentenanță	Suportat de Facebook, cu o comunitate foarte mare, <i>open-source</i>	Suportat de Vercel, cu o comunitate foarte mare, <i>open-source</i>
Documentație	Foarte bună, mai ales pentru începători	Devine mult mai ușoară dacă se cunoaște React deja

Tabela 2.2: React vs Next.js

*Cod

Exemplul de mai jos ilustrează asemănarea dintre React și Next.js. Acesta conține cea mai ușoară afișare a textului "Hello World!" posibilă, și se observă faptul că cele două secvențe de cod sunt aproximativ identice.

```

1 function App() {
2   return (
3     <div className="app">
4       Hello World!
5     </div>
6   );
7 }
8 export default App;

```

Listing 2.3: Hello World în React

```

1 export default function Home() {
2   return (
3     <div className="app">
4       Hello World!
5     </div>
6   )
7 }
```

Listing 2.4: Hello World în Next.js

2.3 Tailwind CSS

2.3.1 Comparatie din punct de vedere aspectual

Tailwind CSS nu este mai eficient doar din punct de vedere al aspectului codului, deși este cu siguranță o particularitate importantă.

```

1 .button {
2   display: flex;
3   align-items: center;
4   justify-content: space-between;
5   border-radius: 4px;
6   border: 1px solid;
7   padding: 8px 16px;
8   color: #F97316;
9   transition: all 0.2s ease-in-out;
10 }
11
12 .button:hover {
13   border-color: #F97316;
14 }
15
16 .button > * + * {
17   margin-left: 12px;
18 }
```

Listing 2.5: Stilizare cu CSS

```

1 className="flex items-center space-x-3 rounded border px-4
2 py-2 text-orange-400 transition hover:border-orange-400"
```

Listing 2.6: Aceeași stilizare cu Tailwind CSS

Din exemplele date se observă că *Tailwind CSS* este mai organizat, ocupă mai puțin spațiu și este mai ușor de memorat și scris. Faptul că nu este nevoie să se afle într-un fișier separat de stilizare, face ca tot procesul de stilizare să fie mult intuitiv.

2.3.2 Comparație din punct de vedere al rației dintre cod și stilizare

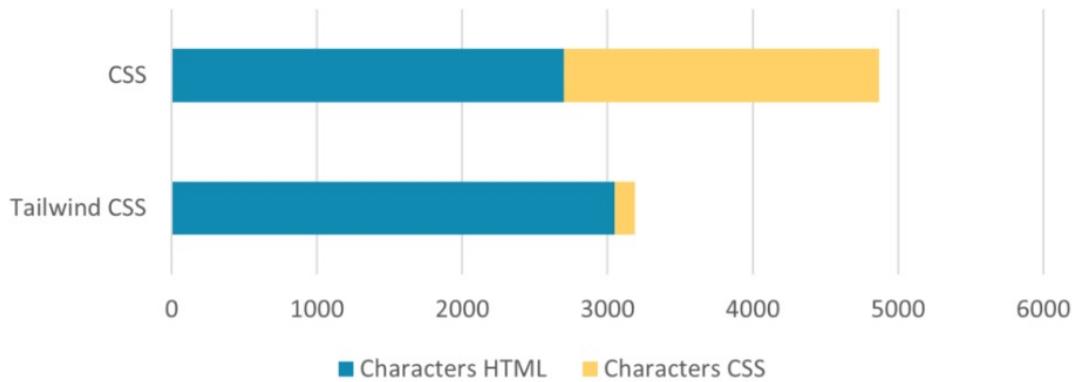


Figura 2.2: CSS vs TailWind CSS (preluată)

Se observă și din exemplele din Anexele 2.5 și 2.6 că sunt necesare mult mai multe caractere pentru a scrie o secvență de cod CSS. Astfel, rația de caractere crește pentru codul HTML și scade pentru TailWind CSS. În schimb, în cazul CSS-ului clasic, rația pare să fie apropiată de 50%-50%.

2.4 Librării externe

2.4.1 dayjs

Day.js este o bibliotecă JavaScript, extrem de ușor de utilizat, care oferă o multitudine de metode utile pentru manipularea și formatarea datelor.

În proiect, o folosesc pentru a afișa data curentă în mai multe contexte:



Figura 2.3: Exemple day.js din aplicație

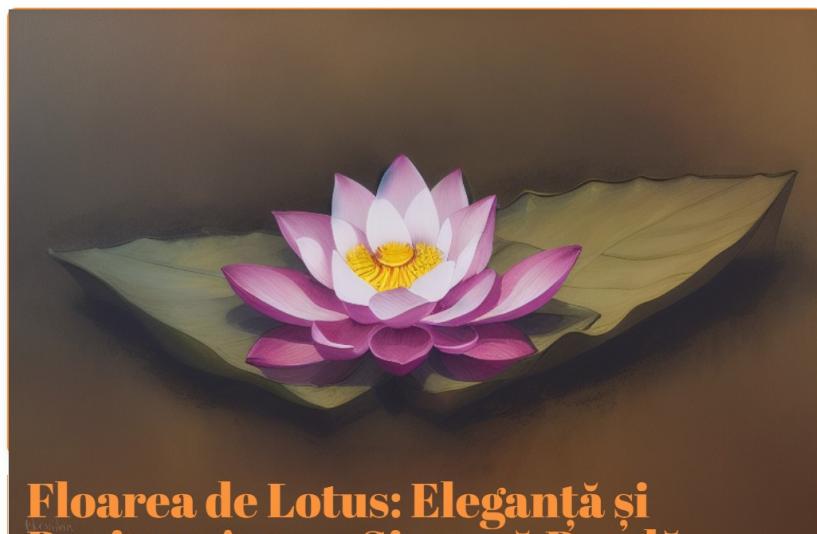
Această bibliotecă îmi permite să afișez o dată, în diverse formate:

- `dayjs().format('YYYY-MM-DD')` - "2023-06-19".
- `dayjs().format('LL')` - "June 19, 2023".
- `dayjs().format('YYYY-MM-DD HH:mm')` "2023-06-19 14:35".
- `dayjs().format('HH:mm:ss')` - "14:35:20".
- `dayjs().format('YYYY-MM-DDTHH:mm:ssZ')` - "2023-06-19 14:35:20+02:00".
- `dayjs().format('D MMMM YYYY, h:mm A')` - "19 iunie 2023, 2:35 PM".

2.4.2 slugify

Slugify este o bibliotecă JavaScript folosită pentru transformarea unui text de orice formă într-un format adecvat pentru URL-uri, cunoscut sub numele de "slug".

În general se folosește *Slugify*, deoarece este foarte util pentru SEO (optimizarea pentru motoarele de căutare) și pentru a asigura că slug-urile sunt formate din litere simple, cifre și liniuțe, fără diacritice. Recunoaștem un slug după faptul că toate cuvintele sunt separate prin liniuțe, inclusiv semnele de punctuație.



Floarea de Lotus: Eleganță și Puritate într-o Singură Petală

Floarea de lotus emana o eleganță subtilă și o puritate uluitoare în fiecare petală. Această minunată floare, cu simboluri profunde și învățăminte spirituale, te va invăța într-o atmosferă de liniste și frumusețe. Clică pe imaginea de mai jos pentru a vedea mai multe despre istoria și semnificația acestei plante unice, care aduce o notă de mister și armonie în grădinile și inimile noastre.

Figura 2.4: Postare: Floarea de Lotus: Eleganță și Puritate într-o Singură Petală

Pentru postarea de mai sus, URL-ul va arăta astfel:

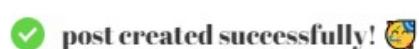
`http://localhost:3000/Floarea-de-Lotus:-Eleganta-si
-Puritate-intr-o-Singura-Petala`

2.4.3 react-hot-toast

React Hot Toast este o bibliotecă JavaScript folosită pentru adăugarea notificărilor tip "toast" în aplicațiile React. Acestea sunt notificări mici și temporare care apar de obicei pe ecran și care furnizează feedback utilizatorilor fără a îintrerupe interacțiunea lor cu interfața.

Astfel, am adăgat câte o astfel de notificare pentru fiecare caz în care trebuie să aștepțăm un răspuns de confirmare a adăugării cu succes a următoarelor proceduri:

- Adăugarea unei postări pe pagina principală.



- Adăugarea unei etichete pentru o postare.



- Adăugarea unui comentariu pe pagina unei postări.



- Adăugarea sau ștergerea în cadrul listei de prieteni



2.5 Concluzie

Acest capitol a detaliat aspectele tehnice ale tehnologiilor utilizate pentru dezvoltarea aplicației și motivul pentru care acestea sunt cele mai potrivite având în vedere

funcționalitățile ca utilizatorii să creeze, să îmbunătățească și să împărtășească desene folosind un sistem de tip blog. *T3 Stack*, compus din 6 tehnologii integrate, oferă o structură solidă și ușor de urmărit, facilitând o comunicare eficientă între *frontend* și *backend*.

Piesa principală, *TypeScript*, un superset al *JavaScript* care aduce avantaje semnificative în dezvoltarea frontend, precum o mai bună gestionare a erorilor și suport pentru tipuri statice și interfețe.

Am comparat de asemenea *React* și *Next.js*, am constatat că *Next.js* este un framework construit pe *React* care aduce îmbunătățiri la nivel de performanță, dar și restricții, precum folosirea exclusivă a *TypeScript* și o structură de foldere mai strictă.

Tailwind CSS a fost ales pentru stilizarea interfeței datorită avantajelor sale în termeni de eficiență a codului, iar librăriile externe precum *dayjs*, *slugify* și *react-hot-toast* au fost integrate pentru manipularea și formatarea datelor, generarea URL-urilor și furnizarea de *feedback* utilizatorului în operațiile pe care le executa.

Așadar, combinația de tehnologii selectate pentru acest proiect nu numai că facilitează implementarea aplicației, dar și îmbunătățește experiența utilizatorului prin funcționalități intuitive și *feedback* instantaneu.

Capitolul 3

Dezvoltarea aplicației web

Acest capitol explorează arhitectura aplicației împreună cu funcționalitățile sale, scenariile de utilizare ale acestora, precum și arhitectura bazei de date.

3.1 Arhitectura Server-Client

Proiectul reflectă principiile fundamentale ale arhitecturii client-server.

tRPC nu este un server propriu-zis și, prin urmare, trebuie să fie găzduit folosind Next.js. tRPC este un *framework* pentru dezvoltarea de aplicații bazate pe API-uri de tip RPC (*Remote Procedure Call*).

În cadrul unei aplicații TypeScript este recomandată utilizarea unui backend tipizat, care permite definirea tipurilor specifice pentru date. În general acest lucru poate reduce erorile și îmbunătăți întreținerea codului. De asemenea, voi putea apela direct metodele unui obiect pe server de la client, fără a fi nevoie de multiple cereri HTTP, aşa cum ar fi necesar în REST.

Cu tRPC, nu ar fi nevoie scrierea manuală a interfețelor pentru fiecare punct de acces al API-ului, aşa cum ar fi necesar în REST. În schimb, pot genera automat aceste interfețe pe baza codului.

În secvența de mai jos este un exemplu de procedură specifică tRPC, ce se bazează pe un model de rutare bazat pe *hook-uri* furnizat de Next.js. `useRouter` este un hook React care oferă acces la obiectul router al Next.js. Acest obiect router permite manipularea și accesul la calea curentă și parametrii din URL. Apoi, se face o cerere de a obține profilul unui utilizator folosind biblioteca tRPC și se preia "username" din URL pentru a face o cerere la server și a obține profilul utilizatorului corespunzător.

Datele profilului utilizatorului sunt apoi stocate în variabila userProfile.

```
1 const router = useRouter();
2 const userProfile = trpc.user.getUserProfile.useQuery(
3     { username: router.query.username as string }
4 );
```

Listing 3.1: client.ts

În partea de server, metoda *getUserProfile* este o procedură care acceptă ca intrare un obiect cu o singură proprietate: *username*, care este un sir de caractere. Această procedură este definită ca o interogare asincronă, adică funcția poate efectua operații care durează un timp mai lung, cum ar fi solicitări de rețea sau interogări ale bazei de date, fără a bloca restul programului.

Metoda *findUnique* a obiectului *user* din *prisma* este folosită pentru a găsi un utilizator unic în baza de date și selectează anumite proprietăți ale acestui utilizator pentru a fi returnate în partea de client.

```
1 export const userRouter = router({
2     getUserProfile: publicProcedure.input(z.object({username: z.string()})).
3         query(async ({ctx:{prisma}, input:{username}}) => {
4             return await prisma?.user.findUnique({
5                 where:{ username:username },
6                 select:{
7                     name:true,
8                     image:true,
9                     id:true,
10                    username:true,
11                },
12            })
13        })
14    })
```

Listing 3.2: server.ts

În cadrul input-ului din exemplul de mai sus, *z* este import de la Zod.

```
import z from "zod";
```

Zod este o bibliotecă de validare a datelor pentru JavaScript și TypeScript. Zod poate verifica dacă datele se potrivesc cu un anumit tip în timpul rulării programului.

3.2 Modelul C4 pentru vizualizarea arhitecturii

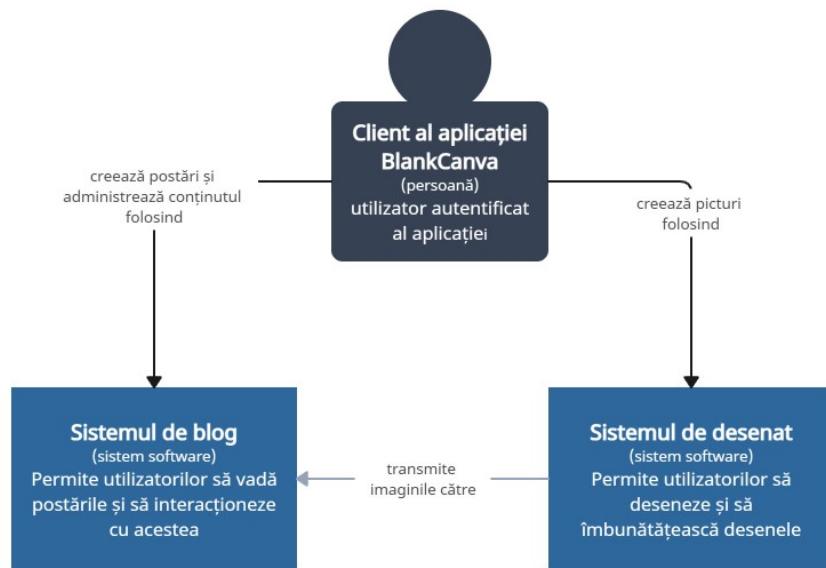


Figura 3.1: Diagramă de context

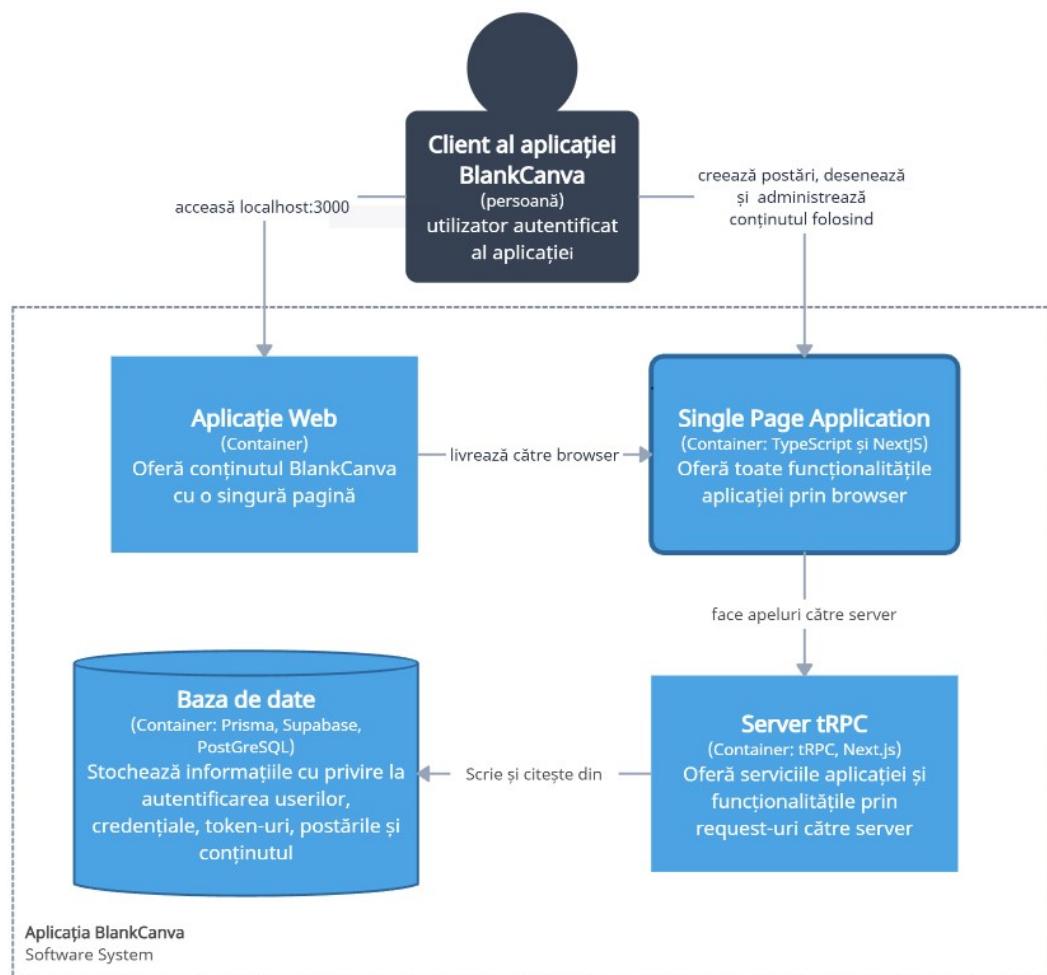


Figura 3.2: Diagramă de container

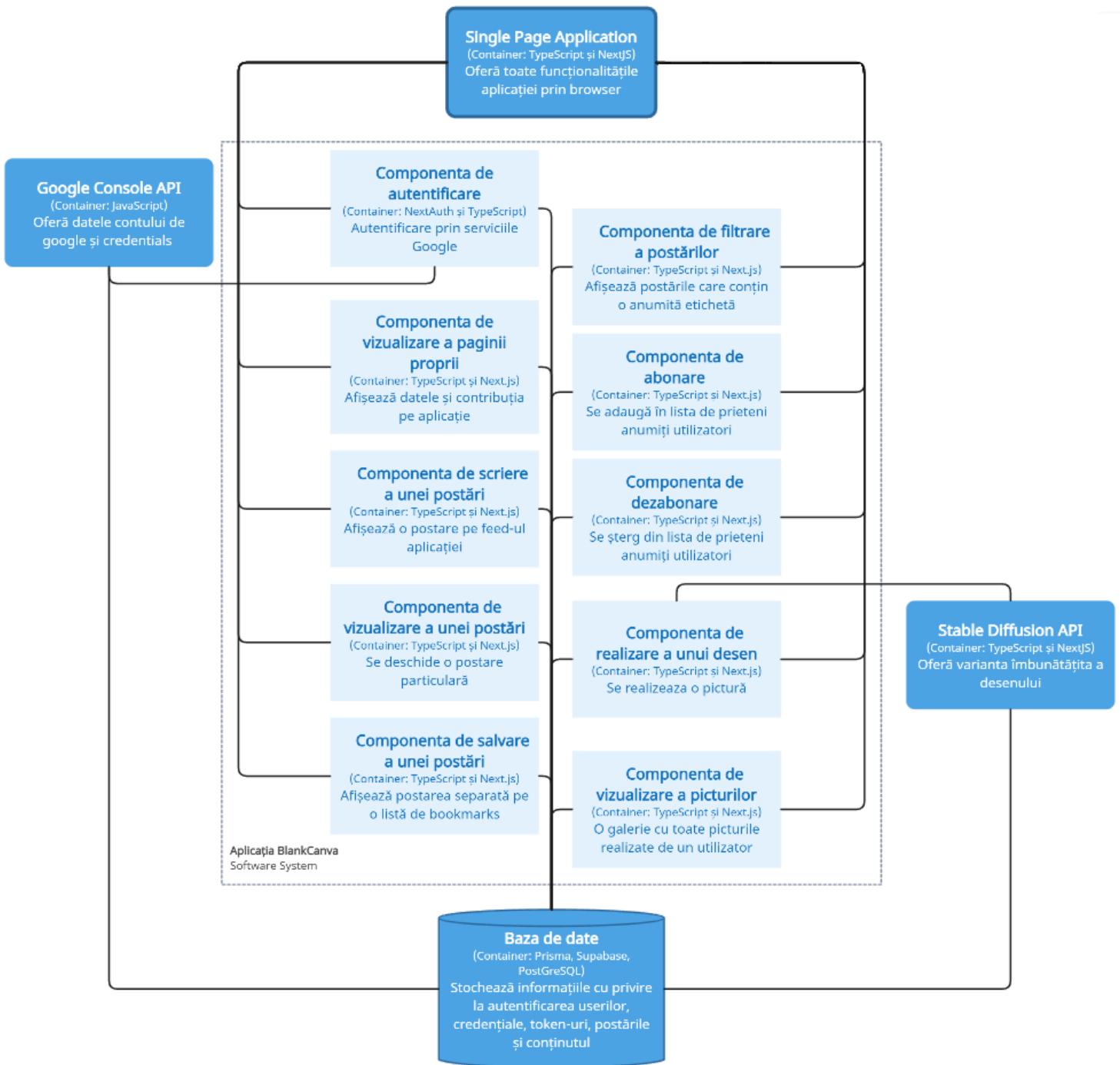


Figura 3.3: Diagramă de componente

3.3 Detalii tehnice ale arhitecturii

Așadar în diagrama Use Case ce urmează, sunt ilustrate sumar funcționalitățile aplicației ca rezumat al diagramelor anterioare, pentru referințe ulterioare:

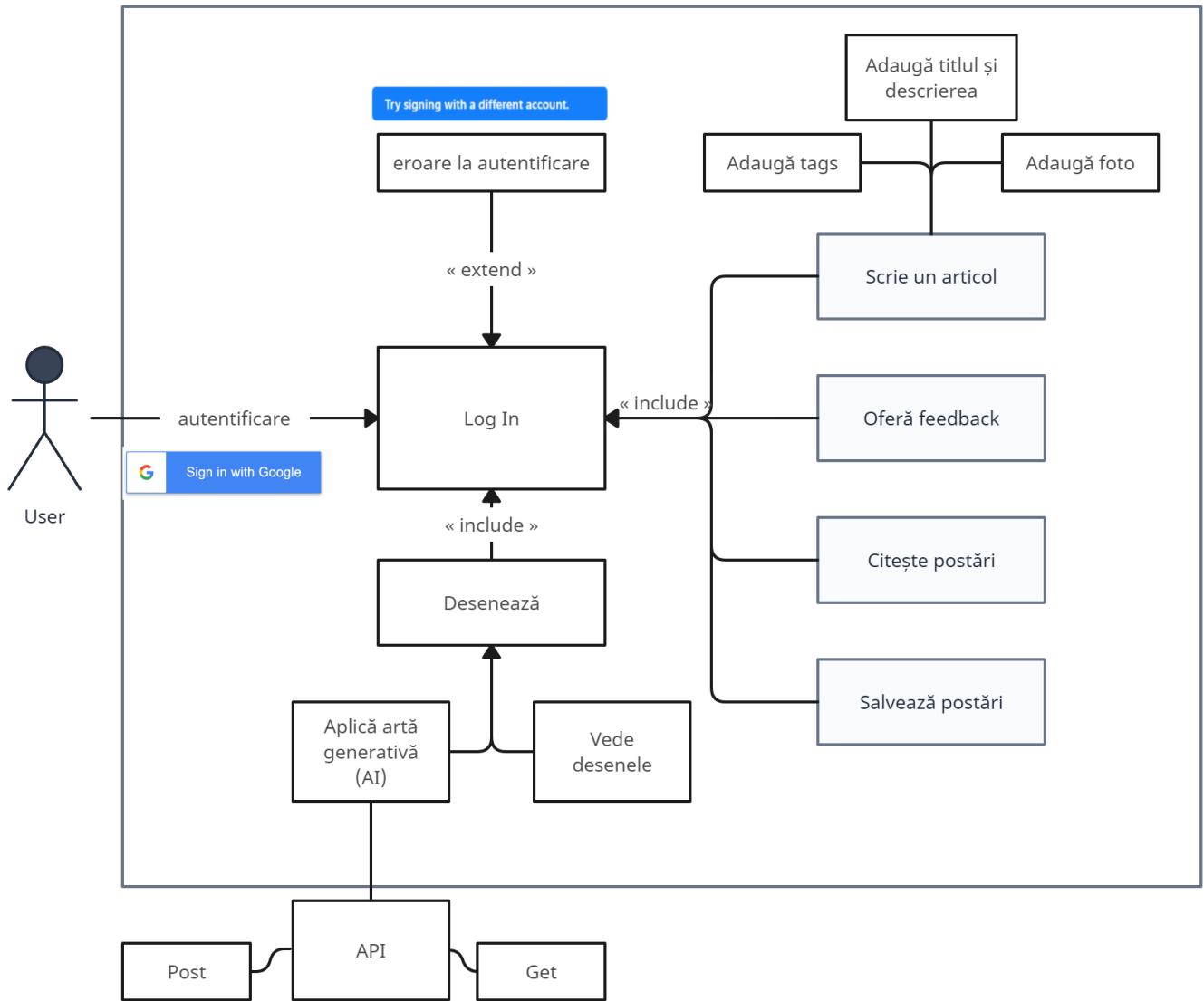


Figura 3.4: Diagramă Use Case

3.3.1 Autentificarea în aplicație

NextAuth.js este conceput pentru a fi ușor de integrat într-o aplicație Next.js. NextAuth.js oferă suport pentru o varietate de furnizori de autentificare, cum ar fi Google, Facebook, Twitter, GitHub.

În proiect, am decis să merg mai departe cu autentificarea prin Google, deoarece oferă o experiență familiară și convenabilă pentru utilizatori, fiind deja o obisnuință autentificarea cu Google pe diverse platforme. De asemenea, Google implementează protocoale și măsuri de securitate solide pentru autentificarea utilizatorilor.

API-urile Google utilizează protocolul OAuth 2.0 pentru autentificare și autori-

zare. Mecanismul general se bazează pe obținerea credențialelor de client OAuth 2.0 din Consola API Google. Apoi, aplicația solicită un token de acces de la Serverul de Autorizare Google, extrage un token din răspuns și trimite tokenul către API-ul Google.

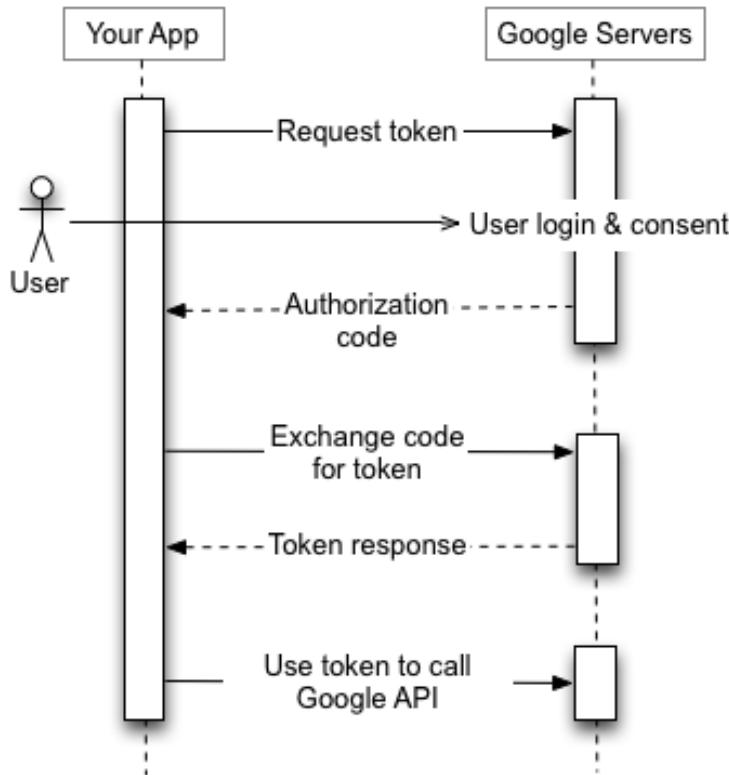


Figura 3.5: Obținerea unui token de acces de la serverele Google (preluată)

NextAuth.js va pune în fișierul prisma, care conține toate modelele de bază de date, câmpurile, relațiile și alte setări de configurare pentru definirea modelului de date al aplicației, cele patru modele necesare pentru funcționare:

- Account: Printre câmpurile comune specifice unui model relational, se remarcă *field*-ul ce **stocă furnizorul de autentificare** asociat cu contul, cel care **stocă tokenul de refresh** pentru cont, cel care **stocă tokenul de acces** pentru cont, cel ce **stocă marcajul de timp la care tokenul de acces expiră**, și cel care stocă **tokenul ID asociat cu contul**, starea sesiunii asociate cu acesta. Câmpul User este un câmp de relație care indică că acest cont este asociat cu entitatea User.

```

model Account {
    id          String  @id @default(cuid())
    userId      String
    type        String
    provider    String
    providerAccountId String
    refresh_token  String? @db.Text
    accessToken   String? @db.Text
    expires_at    Int?
    token_type    String?
    scope         String?
    id_token      String? @db.Text
    session_state String?
    user          User    @relation(fields: [userId], references: [id], onDelete: Cascade)

    @unique([provider, providerAccountId])
    @@index([userId])
}

```

Figura 3.6: Modelul Prisma pentru *Account*

- Session: reprezintă o sesiune în cadrul aplicației, iar scopul acestui model este de a stoca informațiile referitoare la sesiunea unui utilizator.

```

model Session {
    id          String  @id @default(cuid())
    sessionToken String  @unique
    userId      String
    expires     DateTime
    user        User    @relation(fields: [userId], references: [id], onDelete: Cascade)

    @@index([userId])
}

```

Figura 3.7: Modelul Prisma pentru *Session*

- User: reprezintă entitatea de utilizator în aplicație. Google furnizează următoarele informații legate de utilizator: numele, emailul, data verificării emailului la autentificare și fotografia de profil.

Se remarcă și *accounts*, acesta este un câmp relational care indică că utilizatorul poate avea mai multe conturi asociate cu el, și *sessions*, acesta este un câmp relational care indică că utilizatorul poate avea mai multe sesiuni asociate cu el.

```

providers: [
  GoogleProvider({
    clientId: env.GOOGLE_CLIENT_ID,
    clientSecret: env.GOOGLE_CLIENT_SECRET,
    profile(profile){
      return{
        id: profile.sub,
        name: profile.name,
        image: profile.image,
        email: profile.email,
        username: generateUsername(profile.name)
      }
    },
  }),
]

model User {
  id      String     @id @default(cuid())
  name    String?
  username String @unique
  email   String?   @unique
  emailVerified DateTime?
  image   String?
  accounts Account[]
  sessions Session[]
}

```

Figura 3.8: Datele furnizate de Google Console API și Modelul Prisma pentru *User*

- VerificationToken: Scopul acestui model este de a stoca informațiile asociate cu un token de verificare, cum ar fi identificatorul, valoarea tokenului și data expirării. Acest model poate fi utilizat, de exemplu, pentru a verifica și confirma adresele de email ale utilizatorilor.

```

model VerificationToken {
  identifier String
  token      String   @unique
  expires    DateTime

  @@unique([identifier, token])
}

```

Figura 3.9: Modelul Prisma pentru *Verification Token*

3.3.2 Arhitectura bazei de date

Supabase este o platformă *open-source* care furnizează o bază de date relațională și o infrastructură de backend pentru dezvoltarea aplicațiilor. Se bazează pe PostgreSQL și oferă funcționalități complete de bază de date, autentificare, autorizare și gestionare a sesiunilor.

Voi menționa câteva dintre atrbutele pozitive ale acestei platforme:

- actualizează instant în interfață în momentul în care în interiorul bazei de date se produc modificări, fără a trebui să fie dat refresh la pagină

- deține un sistem de autentificare și autorizare dezvoltat, astfel încât manevrarea bazei de date să nu fie făcută în condiții de risc
- permite gestionarea autentificării utilizatorilor și accesul securizat la resursele aplicației
- permite gestionarea și urmărirea sesiunilor utilizatorilor care s-au conectat

Atunci când am realizat instanța locală de Supabase, aş fi putut alege între bazele de date PostgreSQL și MySQL. Mai departe, am ales PostgreSQL fiind mai robust și avansat, aşa cum descrie și următorul tabel:

Proprietate	MySQL	PostgreSQL
Tipul de bază de date	Relațională	Obiect-relațională
Suportă CASCADE	Nu	Da
Complexitatea procedurilor	Sintaxele SQL și procedurile stocate	Proceduri avansate și stocate
Tipul de index suportat	Arbore de căutare binar (B-Tree)	Multe, inclusiv Hash
Criptare între client și server	TLS	SSL
Suport pentru XML	Nu	Da
Suport pentru moștenirea tabelelor	Nu	Da
Suport pentru tipuri de date avansate	Nu	Da – date definite de utilizator

Tabela 3.1: PostgreSQL vs MySQL

În baza mea de date se găsesc 11 tabele, dintre care există două tabele suplimentare, **_UserFollows** (folosită pentru a gestiona relația de urmărire (*follow*) între utilizatori) și **_PostToTag** (folosită pentru a gestiona relația dintre Post și Tag), iar aceste tabele sunt generate automat de către Prisma (Many-to-Many).

În figura următoare pot fi observate atât restul de tabele din proiect, cât și felul în care acestea interacționează în cadrul aplicației:

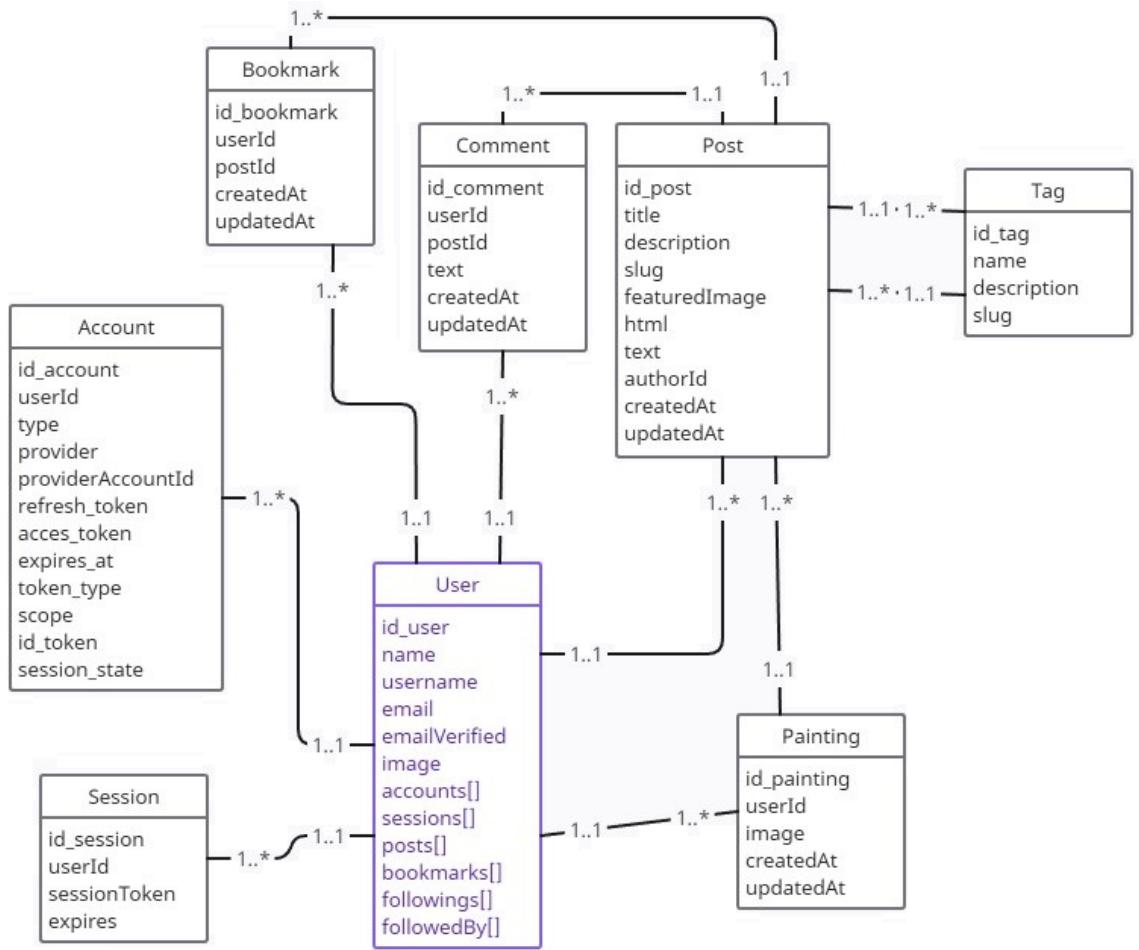


Figura 3.10: Arhitectura bazei de date

- User are o relație de tip unu-la-multi cu modelele *Account*, *Session*, *Post*, *Comment*, *Like*, *Bookmark* și *Painting*, indicând că **un utilizator poate avea mai multe conturi, sesiuni, postări, comentarii, aprecieri, salvări ale postărilor și picturi**.
- Post reprezintă o postare în aplicație, și are o relație de tip multi-la-unu cu *User*, indicând că **fiecare postare este creată de un singur utilizator**. De asemenea, Post are o relație de unu-la-multi cu *Comment*, *Like*, *Bookmark*, sugerând că **o postare poate avea mai multe comentarii, aprecieri și marcaje**. Post este, de asemenea, legat de Tag printr-o relație multi-la-multi, sugerând că **o postare poate avea mai multe etichete asociate**.
- Modelele *Comment* și *Bookmark* (reprezintă salvarea unei postări din cadrul aplicației într-o listă personală), au ambele o relație de tip multi-la-unu cu *User* și *Post*, indicând că **fiecare comentariu sau bookmark este asociat unui singur utilizator și unei singure postări**.

- Tag reprezintă o etichetă care poate fi aplicată unei postări și are o relație de tip multi-la-multi cu *Post*, indicând că **o etichetă poate fi asociată mai multor postări**.
- Painting reprezintă o pictură creată de un utilizator și are o relație de tip multi-la-unu cu *User*, sugerând că **fiecare pictură este asociată unui singur utilizator**.

3.4 Virtualizarea aplicației la nivel de sistem de operare

Virtualizarea unei aplicații la nivel de sistem de operare este realizată prin intermediul **tehnologiei de containere**.

Practic, ne ajută în dezvoltarea unei aplicații să nu întâmpinăm probleme atunci când ne mutăm aplicația, spre exemplu pe alt sistem de operare, iar acest lucru se datorează faptului că un container **își izolează software-ul de mediul înconjurător** și asigură că funcționează uniform, indiferent de diferențele dintre sistemele de dezvoltare și producție.

În proiectul meu, am decis să merg mai departe cu Docker, fiind unul dintre cele mai populare unelte pentru a realiza acest lucru. Docker este o platformă de containerizare care permite dezvoltatorilor să creeze, să implementeze și să ruleze aplicații în mod convenabil cu ajutorul containerelor.

Docker permite separarea aplicațiilor de infrastructură, astfel încât să fie posibilă livrarea software-ului rapid. Profitând de metodologiile Docker pentru expediere, testare și implementare rapidă a codului, se reduce semnificativ întârzierea dintre scrierea codului și rularea acestuia în producție. Funcționează astfel:

- Se dezvoltă aplicația și componentele sale auxiliare folosind containere. Containerul devine unitatea pentru distribuirea și testarea aplicației dvs.
- Se implementează aplicația în mediul propriu de producție la fel ca în mod normal, în timp ce aceasta se află în container.

Obiectele Docker

- Imaginea

O imagine este un şablon cu instrucțiuni pentru crearea unui container Docker. Adesea, o imagine se bazează pe o altă imagine, cu personalizări adiționale,

dar se pot crea și propriile imagini prin crearea un Dockerfile pentru definirea pasilor necesari pentru a crea imaginea respectivă și a o rula. Fiecare instrucțiune dintr-un Dockerfile creează un strat în imagine. Când schimbați Dockerfile și reconstruiți imaginea, doar acele straturi care s-au schimbat sunt reconstruite.

- Containerul

Un container este o unitate standard de software care împachetează codul și dependențele sale, își izolează software-ul de mediul înconjurător și asigură că funcționează uniform. Un container este o instanță executabilă a unei imagini. Printre operațiile pe care le putem realiza asupra unui container, se enumera și posibilitatea de a conecta un container la una sau mai multe rețele, crea o nouă imagine pe baza stării sale curente, muta, șterge, porni.

Pașii pentru virtualizarea prin Docker

1. Prima etapă este crearea unei imagini Docker pentru aplicație, care va conține toate dependențele necesare pentru a o rula. Acest lucru se face prin **fișierul Dockerfile**. Se utilizează comanda docker build pentru a crea o imagine Docker.
2. Se utilizează comanda docker run pentru a lansa un container pe baza acelei imagini. Acest container va rula aplicația într-un mediu izolat.
3. Acum se poate distribui imaginea Docker pe mai multe medii de lucru.

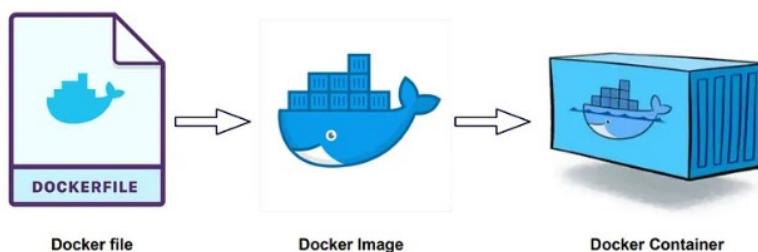


Figura 3.11: Schema de funcționare Docker (preluată)

3.5 Concluzie

Acest capitol detaliază dezvoltarea unei aplicații web, stabilind cum arată arhitectura aplicației, ce presupune fiecare funcționalitate și o analiză amplă asupra arhitecturii

bazei de date. Aplicația detine o arhitectură client-server iar framework tRPC, care a susținut întreaga arhitectură, a adus avantaje în ceea ce privește întreținerea codului și eficiența apelurilor către server.

Se descrie și procesul de autentificare, cu accent pe utilizarea NextAuth.js și integrarea cu Google, precum și implementarea OAuth 2.0. Datorită acestui tip de autentificare, utilizatorii ajung mai ușor la funcționalitățile proiectului. Deși depinde de intențiile fiecărui programator de a alege metoda de autentificare, am decis că Google este foarte explicit și sigur în ceea ce privește felul în care manevrează datele utilizatorilor.

În secțiunea privind arhitectura bazei de date se prezintă Supabase, un serviciu open-source bazat pe PostgreSQL, care ajută prin actualizările în timp real.

În final, capitolul tratează virtualizarea la nivel de sistem de operare folosind tehnologia de containere, cu scopul de a izola software-ul și astfel facilitând compatibilitatea între diferite sisteme de operare.

Capitolul 4

Componența de desen digital

4.1 Proiectarea aplicației de desenat

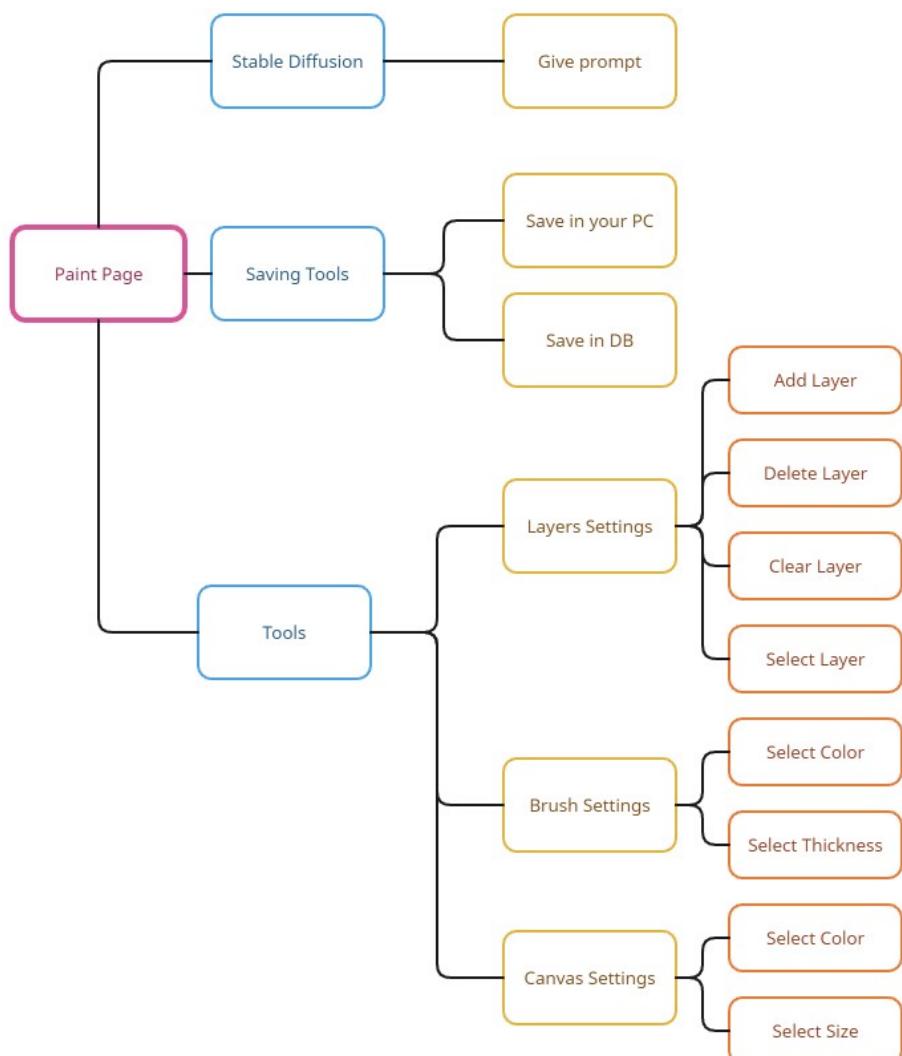


Figura 4.1: Funcționalitățile componenței de desen

Ideea proiectului este ca utilizatorii să își dezvăluie exclusiv lucrările realizate cu această componentă. Totuși, este la alegerea utilizatorului dacă acesta dorește să posteze picturile pe care le-a făcut, el are acces la această componentă de desen indiferent, și poate oricând să vadă lucrările anterioare sau să creeze altceva nou.

Funcționalitățile acestei componente, conforme cu Figura 3.1 sunt:

- UTELTE UTILE ÎN DESEN

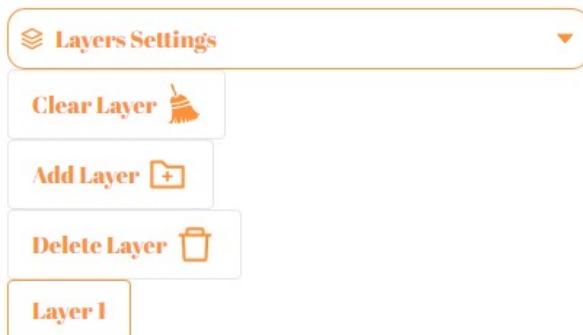


Figura 4.2: Interfața pentru setările ce țin de stratificarea desenului

- **Adaugă layer** - Această funcționalitate permite utilizatorilor să adauge un nou strat în cadrul desenului. Prin adăugarea de noi layere, se poate lucra în mod individual pe fiecare strat, fără a afecta restul desenului, din caz că se produce o greșeală în desen.
- **Curăță layer** - Această funcționalitate permite utilizatorilor să curețe desenul de pe suprafața unui layer. Este utilă în cazul în care se dorește îndepărțarea elementelor de pe un layer, dar pentru desen este necesar un strat în plus și ar fi ineficient să ștergi întreg layer-ul doar pentru adăugarea lui la loc curat.
- **Sterge layer** - Această funcționalitate permite utilizatorilor să eliminate definitiv un layer, aceasta îndepărtează complet acest strat din cadrul desenului, și este util în cazul în care un layer nu mai este necesar sau dacă utilizatorul dorește să înlăture anumite elemente din desen.
- **Selectează layer** - Această funcționalitate permite utilizatorilor să selecteze un anumit layer pe care doresc să lucreze, pentru a nu afecta restul componentelor din spatele layer-ului respectiv.



Figura 4.3: Interfața pentru setările ce țin de pensulă

- **Selectează culoarea pensulei** - Această funcționalitate permite utilizatorilor să aleagă o anumită culoare a pensulei.
- **Selectează grosimea pensulei** - Această funcționalitate permite utilizatorilor să regleze grosimea pensulei, oferind astfel flexibilitate în crearea de linii de diferite dimensiuni.

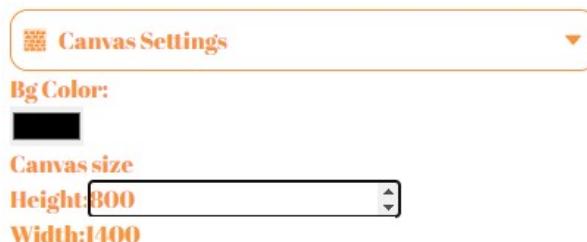


Figura 4.4: Interfața pentru setările ce țin de fundal și pânză

- **Selectează culoarea de fundal** - Această funcționalitate permite utilizatorilor să aleagă culoarea de fundal a desenului.
- **Selectează dimensiunea pânzei** - Această funcționalitate permite utilizatorilor să selecteze dimensiunea pânzei pe care se lucrează.

- Butoane de salvat

Acestea apar pe ecran în momentul în care utilizatorul aplică filtrul de inteligență artificială peste desenul său, și primește un răspuns.



Figura 4.5: Interfața pentru salvarea desenului

- **Salvează în aplicație** - Această funcționalitate permite utilizatorilor să păstreze desene direct în cadrul aplicației.
- **Salvează pe calculatorul propriu** - Această funcționalitate permite utilizatorilor să păstreze desenele pe dispozitivele lor, iar acestea vor fi descărcate pe calculator, permitând astfel păstrarea, partajarea sau tipărirea acestora în funcție de preferințele personale.

4.2 Implementarea aplicației de desenat

4.2.1 Planșa de desen

Pânza funcțională pe care poți desena este implementată cu ajutorul **HTML Canvas Graphics**.

HTML canvas este un element utilizat pentru a desena grafică pe o pagină web, iar funcționalitatea îi este dată atât JavaScript cât și TypeScript.

Elementul `<canvas>` este un standard HTML5, și reprezintă un container pentru grafică, iar pentru a desena este necesar un script. Acesta trebuie să aibă un atribut *id*, *width* și *height*. În mod implicit, elementul `<canvas>` nu are nicio bordură și niciun conținut, aşa cum se observă și în Figura 3.6 de mai jos, unde este adăugat un *border* negru pentru a putea fi vizualizat.

Ulterior se adaugă elementele de TypeScript ce vor permite desenarea asupra planșei, și gestionarea interacțiunii cu mouse-ul pentru a desena pe canvas:

- **mousedown**: Acest eveniment este declanșat când se apasă un buton al mouse-ului. De obicei, se inițiază procesul de desenare în acest moment.
- **mousemove**: Acest eveniment este declanșat atunci când mouse-ul este mișcat. În timpul desenării, poziția curentă a mouse-ului este urmărită pentru a actualiza desenul.
- **mouseup**: Acest eveniment este declanșat când un buton al mouse-ului este eliberat. Aceasta este momentul în care de obicei se încheie procesul de desenare.

Acestea se regăsesc și în aplicația mea, aşa cum se poate observa în codul citat de mai jos. Ulterior, am adăugat și elemente de stilizare, și funcționalitatea de stratificare a desenului.

```

1 <canvas
2     key={i}
3     ref={ref}
4     width={canvasWidth}
5     height={canvasHeight}
6     onMouseDown={handleMouseDown}
7     onMouseUp={handleMouseUp}
8     onMouseMove={draw}
9 />

```

Listing 4.1: Elementul <canvas> în aplicație

Un exemplu al implementării unei astfel de funcții este handleMouseDown, care este atașat mai jos. Funcția primește un argument e, care este evenimentul de mouse care a declanșat apelul funcției și obiectul care conține poziția cursorului mouse-ului pe ecran la momentul apăsării butonului.

```

1 const handleMouseDown = (e: React.MouseEvent<HTMLCanvasElement>) => {
2     setIsDrawing(true);
3     draw(e);
4 }

```

Listing 4.2: Elementul <canvas> în aplicație

Pentru a înțelege mai departe și cum se realizează desenarea efectivă, am atașat și funcția responsabilă de trasat în aplicație.

În această funcție se obține referința la *layer*-ul curent pe care se desenează, și se verifică dacă utilizatorul este în procesul de a desena, iar apoi se obține contextul de desenare 2D al canvas-ului prin apelarea metodei `getContext ("2d")`.

Se obține poziția canvas-ului peste care utilizatorul desenează în momentul actual prin apelarea metodei `getBoundingClientRect()`. Acest lucru este necesar pentru a calcula coordonatele corecte ale punctului de desenare. Coordonatele cursorului sunt date de `e.clientX` și `e.clientY`, dar se scade poziția canvas-ului pentru a obține coordonatele relative la canvas, nu la întregul ecran, iar apoi vom observa că linia desenată apare pe ecran, dat fiind apelarea metodei `context.stroke()`.

```

1 const draw = (e: React.MouseEvent<HTMLCanvasElement>) => {
2     const canvas = canvasRefs[activeLayer]!.current;
3     if (canvas && isDrawing) {

```

```

4   const context = canvas.getContext("2d");
5
6   if (context) {
7     const rect = canvas.getBoundingClientRect();
8     context.lineTo(e.clientX - rect.left, e.clientY - rect.top);
9     context.stroke();
10  }
11 }

```

Listing 4.3: Elementul <canvas> în aplicație

4.2.2 Salvarea planșei de desen

Pentru salvarea planșei de desen, am convertit imaginea într-un string. Acest string conține toate datele dintr-o imagine, pixel cu pixel. Pentru această convertire, m-am ajutat de Base64.

Base64 este deosebit de frecvent pe World Wide Web, unde una dintre utilizările sale este capacitatea de a încorpora fișiere de imagini sau alte resurse binare în cadrul activelor textuale, cum ar fi fișierele HTML și CSS.

Codificare

În proiectul meu, am convertit canvas-ul, ca o colecție de toate straturile cu desene, unite într-o singură fotografie. M-am asigurat ca toate layerele să fie conectate între ele și ca rezultatul final pe care o voi converti să fie un obiect întreg, nu o colecție separată de layere.

Pentru convertirea unei imagini într-un astfel de string, care este de o dimensiune foarte mare, nu este nevoie decât de o linie de cod.

```
const base64 = canvas.toDataURL("image/png");
```

Decodificare

De asemenea, pentru decodarea unui base64 string, este suficient doar să punem în src acest text, și imaginea convertită va apărea instant.

```
<img src='data:image/png;base64,$base64' alt="" />
```

```

data:image/png;base64,iVBORw0KGgoAAAANSUhEUg
AABXgAAAMgCAYAACZBqgXAAAAAXNSR0IArs4c6QAAIA
BJREFUeF7svQnw81V3/f1zLx15s28mffeLBpmCdpLG
AK1LAZDOWSyqZGpsBlcAixQ6oSY1dMTIRUxSsgJJt4gc
Ipg4OTEFIVSIKBBMrYLraQSFXGGIFYDDMyTIRm3/ftjQ
RJ6mj
+P83v/efX3efe231vL59f1aupmn8v53x0377d554+/Qf
EDwIQgAAEIAABCCEAAhCAAAR6IfCcpDOS/sDRvx70+v8
k3S/p/g/KoAMETAABCCEAgNwF76fODAAQgAAEIQAACEIA
ABCAAgXYJ/L6kK9oVf5Lk1yRdPakGhSEAAQhAAAKdE8D
B27mBUQ8CEIAABCQAAQhAAAIQ6JLAxyVd1aVmPqXMqT2
y/j5K1IIABCAGSEI40AdwswoCQEIQAACEIAABCQAAQh
0QOAlSaczpF6w1Af27/cknaqIy/87QzFSN1RkQESBAAQ
gAIFTCDg3YY7vUIAAhCAAQgAAEIQAACEPASsLQESx2
01HWUm0IKD11Q0bR5KbQaXYrbF/um0juPj7J03I1Kb31
4ix3qjtJ9nZwpBAAJTCdwP6fyKztaHjb0h0B
+pGeY7dy3lha1zevt5nbv3brYm/LoAwEIQAACEIBAxz
KVsedYBFpx8MYW3LUcA7fNzomOnLqxJ4G0D75541edF8
:

```



Figura 4.6: Exemplu simplu de cum funcționează decodificarea unui string base64

Această metodă de salvare a unei imagini este, în opinia mea, cea mai potrivită metodă pentru acest proiect, fiind atât de simplu să încorporezi imagini direct în documentele HTML sau CSS. O problemă care ar putea apărea în momentul în care salvarea unei informații pe un alt server, este încălcarea principiilor CORS. Salvarea unui string în loc de fotografia efectivă simplifică structura bazei de date și dispără nevoia de a gestiona fișierele de imagine separate.

De asemenea, fiind un proiect de tip Server-Client, trimiterea imaginilor de la server la client este mai simplă sub formă de siruri de caractere, și în plus există acest avantaj că toate browserele web moderne suportă manipularea și afișarea imaginilor codificate Base64.

4.3 Concluzie

Capitolul prezentat descrie implementarea componentei de desen digital în cadrul aplicației. Aceste funcționalități sunt intitulate sugestiv, iar acestea sunt adăugarea, ștergerea sau curățarea unor straturi, alegerea culorii pensulei și a fundalului, reglarea grosimii pensulei și selectarea dimensiunii pânzei. Opțiunile de salvare în aplicație sau pe dispozitivul propriu facilitează accesul și utilizarea lucrărilor create, iar disponerea butoanelor de salvare după ce apare imaginea răspuns de la Stable Diffusion, ajută să se păstreze un stil general în ceea ce privește postările. De asemenea, imaginile sunt impresionante, și eventual se pot păstra pe dispozitivul propriu pentru observarea lor mai în detaliu.

Implementarea planșei de desen s-a realizat folosind HTML Canvas Graphics, iar

procesul de desenare este gestionat printr-o serie de evenimente, precum mousedown, mousemove și mouseup, iar în ceea ce privește statusul desenului, am implementat și o serie de funcții specifice care permit urmărirea și actualizarea desenului în timp real.

În cele din urmă, pentru a asigura o stocare eficientă, aplicația convertește imaginiile într-un sir de caractere folosind codificarea Base64. Acest proces simplifică structura bazei de date și facilitează trimitera imaginilor de la server la client, fără a se încalca principiile CORS.

Prin toate aceste caracteristici, componenta de desen digital oferă utilizatorilor un mediu creativ și interactiv pentru a-și exprima talentele și ideile artistice, garantând în același timp stocarea și accesul ușor la lucrările lor.

Capitolul 5

Inteligenta artificială în arta digitală

Inteligenta artificială redefineste actual în mod permanent limitele artiștilor și persoanelor ce concep lucrările creative. Picturile realizate prin intermediul inteligenței artificiale sunt complexe și nu există îndoială că aceasta a devenit o prezență permanentă în domeniul artei. Datorită progreselor tehnologice majore în domeniul sistemelor AI generative, oricine poate produce imagini uimitoare, doar descriind ceea ce dorește să vadă.

5.1 Rețele Generative Adversariale

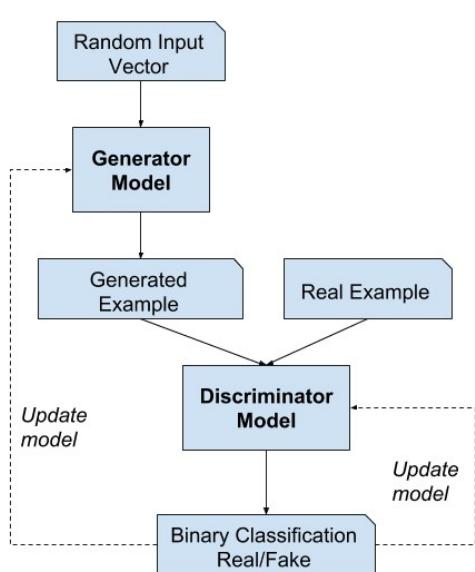
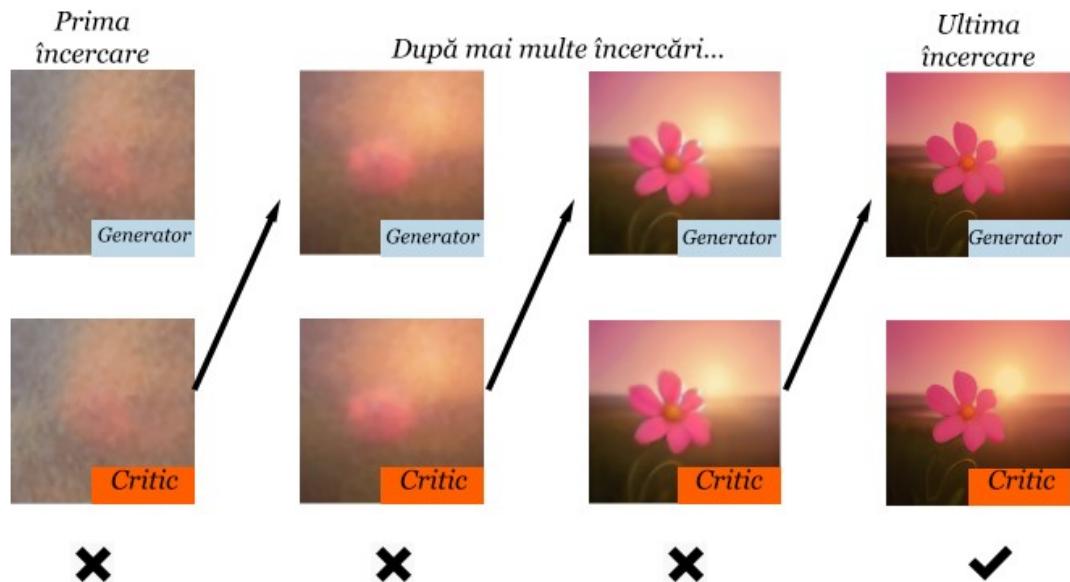


Figura 5.1: Schemă generală (preluat)

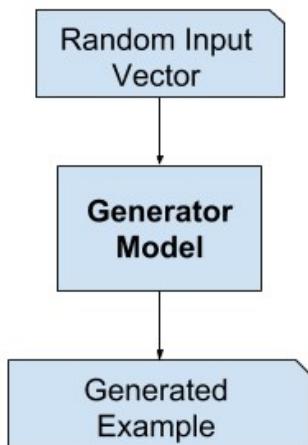
Un mod standard de generare al imaginilor este bazat pe rețele generative adversariale. Rețelele Generative Adversariale (GANs) sunt una dintre cele mai interesante idei din domeniul informaticii în prezent, și presupune că două modele sunt antrenate simultan prin intermediul unui proces adversarial, și au ca rezultat imaginile pe care le primim noi, generate de AI.

- **Generator ("artistul"):** Modelul folosit pentru a genera exemple plauzibile noi din domeniul problemei, el învață să creeze imagini noi care par reale

- **Discriminator ("criticul de artă")**: Modelul folosit pentru a clasifica exemplele ca fiind reale (din domeniu) sau false (generate)



Generator



Modelul generator primește la intrare un vector de lungime fixă, extras aleatoriu dintr-o distribuție gaussiană și folosit pentru a iniția procesul de generare. Acesta va genera o moștră în domeniul respectiv.

După antrenare, punctele din acest spațiu multidimensional de vectori vor corespunde punctelor din domeniul problemei, formând o reprezentare comprimată a distribuției datelor.

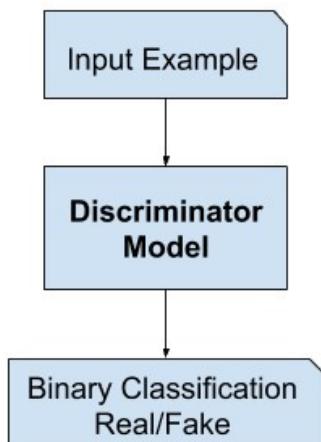
În cazul GAN-urilor, modelul generator atribuie semnificație punctelor dintr-un spațiu latent ales,

Figura 5.2: Generator
(preluat)

astfel încât noi puncte extrase din spațiul latent pot fi furnizate modelului generator ca intrare și pot fi utilizate pentru a genera exemple noi și diferite.

După antrenare, modelul generator este păstrat și utilizat pentru a genera mostre noi.

Discriminator



Modelul discriminator primește ca intrare un exemplu din domeniu (real sau generat) și prezice o etichetă de clasă binară, fie reală, fie falsă (generată).

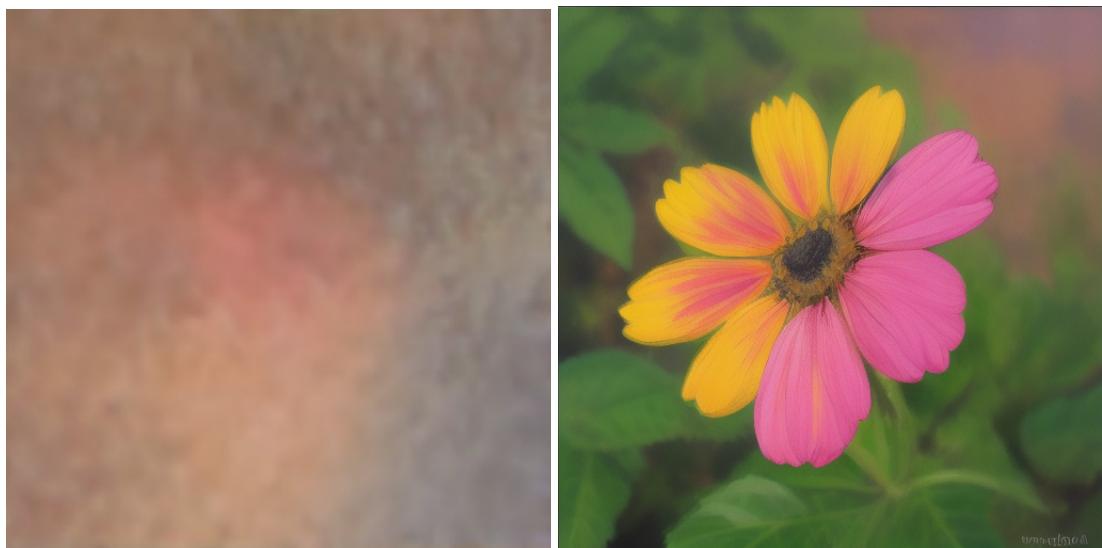
Exemplul real provine din setul de date de antrenament. Exemplele generate sunt rezultatul modelului generator.

Discriminatorul este un model de clasificare normal. După procesul de antrenare, modelul discriminator este eliminat deoarece suntem interesați de generator.

Figura 5.3: Discriminator

(preluat)

Din punct de vedere vizual, acest proces se afișează sub formă de *noise* ce devine progresiv mai ușor de intuit, până se afișează imaginea finală.



5.2 Difuziune

Așadar, modelele generative sunt o clasă de modele de învățare automată care pot genera date noi pe baza datelor de antrenament.

Modelele de difuzie sunt tipuri de modele generative care utilizează procese probabilistice pentru a transforma datele dintr-o distribuție simplă într-o distribuție șițintă mai complexă. Aceste modele rafinează în mod iterativ distribuția aleatoare

inițială, unde în fiecare pas elimină zgomotul din date și, în cele din urmă, creează o moștă realistă de date.

Ulterior, o rețea neurală este antrenată să recupereze datele originale prin inversarea procesului de adăugare a zgomotului. Prin a putea modela procesul invers, putem genera date noi. Aceasta este denumit procesul de difuzie inversă sau, în general, procesul de eșantionare al unui model generativ. Algoritmii de eșantionare permit explorarea eficientă a parametrilor, facilitând convergența mai rapidă la distribuția dorită. Prin reducerea varianței eșantioanelor, algoritmii de eșantionare pot produce estimări ale distribuției țintă mai precise. Algoritmii de eșantionare sunt, de asemenea, robusti și pot reacționa rapid la schimbările în distribuție, ceea ce este util în situațiile în care distribuția se modifică constant.

Modelele de difuzie latentă aplică procesul de difuzie într-un spațiu latent mai mic pentru eficiență computatională utilizând un autoencoder variational pentru redimensionarea în sus și în jos.

Modelele bazate pe scoruri aplică, de asemenea, o secvență de perturbări de zgomot asupra imaginii originale. Însă acestea sunt antrenate folosind potrivirea scorului și dinamica Langevin. Cu toate acestea, ele ajung la un obiectiv similar.

5.2.1 Comparație între GAN și difuziune

	Avantaje	Dezavantaje
GAN	Pot sintetiza date de înaltă calitate, inclusiv imagini și sunete	Antrenarea poate fi dificilă
	Sunt relativ mai rapide în procesul de antrenare	Sensibile la alegerea hiperparametrilor
	Pot utiliza o varietate de funcții de pierdere	Mode collapse
Modelul de difuzie	Pot genera eșantioane de date de înaltă calitate, cu detalii specifice	Antrenarea poate fi lentă și costisitoare
	Antrenate utilizând estimarea maximă probabilități	Necesită o cantitate mare de date pentru antrenare

Succesul este datorat faptului că modelele de difuziune sunt antrenate pe miliarde de imagini, astfel sunt capabil să creeze, să modifice imagini deja existente pe baza unei descrieri textuale, și în prezent poate chiar să îmbunătățească imaginile cu rezoluție redusă, adăugând detalii și fiind insesizabil că imaginea este modificată de

5.3 Stable Diffusion

Lansarea modelului Stable Diffusion reprezintă un punct de referință clar în această evoluție, deoarece pune la dispoziția publicului un model cu performanțe ridicate, atât prin calitatea superioară a imaginilor generate, cât și la viteza mare de procesare și la cerințele relativ reduse de resurse și memorie, comparativ cu alte modele existente.

Stable Diffusion este un model de învățare profundă de tip text-to-imagine și image-to-image, lansat în 2022. *Stable Diffusion* este folosit în principal pentru a genera imagini detaliate bazate pe descrieri textuale, dar poate fi utilizat și pentru alte sarcini, precum reconstituirea și extinderea imaginilor.

Modelul a fost dezvoltat de start-up-ul *Stability AI*, în colaborare cu numeroși cercetători academici și organizații non-profit. Codul său este public și poate fi rulat pe majoritatea echipamentelor echipate cu un GPU modest cu cel puțin 8 GB de memorie video.

Aceste modele pot utiliza o cantitate destul de mare de resurse computaționale, în special atunci când lucrează cu imagini de rezoluție înaltă, astfel că pot apărea situații în care utilizatorii se confruntă cu erori de memorie *CUDA* insuficientă.

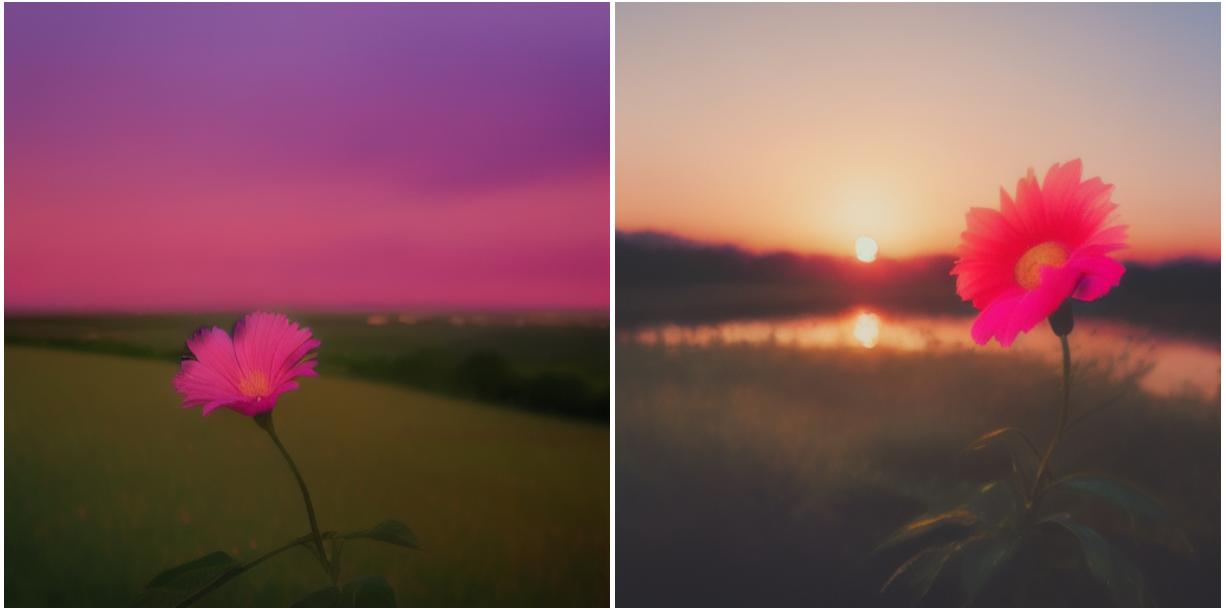
Pentru rularea într-o sesiune pe calculatorul personal, este necesară versiunea Python 3.10.6, instalarea git-ului și clonarea repository-ului ce conține toate fișierele necesare, urmând să se deschidă în browser interfață.

5.3.1 Prompt

Prompt-ul este textul după care se creează imaginea. Un prompt este un instrument foarte important pentru dezvoltarea aplicațiilor AI, pentru că ajută la furnizarea informațiilor necesare pentru antrenarea modelelor dorite, pot ajuta utilizatorii să facă aplicația de AI să înțeleagă mai bine contextul, și să ofere prin prompt informații suplimentare pentru un rezultat mai bun.

Legat de crearea unui prompt, este important să ne asigurăm că oferim suficiente informații clare pentru antrenare. Mai jos este ilustrat un caz în care dăm două prompturi ce exprimă același lucru, însă sub grade de detaliere diferite, iar imaginile sunt diferite în acord cu diferența din prompt: *"Pink flower sunset"* vs *"Vibrant pink flower*

gracefully positioned in the foreground, while being bathed in the warm hues of a breathtaking sunset in the background”.



5.3.2 Negative Prompt

Un prompt negativ se aseamănă la caracteristici cu cel simplu, însă trebuie să ținem cont de faptul că orice instrucțiune o redăm în cadrul unui prompt negativ, imaginea se va crea având în vedere să evite elementele pasate.

Deci, prompt-urile negative sunt utilizate pentru a specifica ceea ce nu trebuie să fie rezultatul, sau pentru a indica modelului să nu ofere în rezultat anumite trăsături.

Utilizarea lor este utilă atunci când se dorește controlul mai precis asupra rezultatelor generate, iar acest lucru este ilustrat în exemplul de mai jos, unde pentru prima fotografie am inserat și un prompt negativ, acesta fiind "*orange petals*".



5.3.3 Modelul ales

Un desen poate lua multiple forme și poate aparține mai multor categorii, însă, tipul de desen ce reprezintă cel mai bine această aplicație este pictura. Am căutat un model care să realizeze din niște simple desene, reale picturi, sau un stil asemănător, care să pară cât mai uman.

Astfel, modelul ***Impressionism (Oil painting)*** obținut de pe CivitAI, al cărui autor a antrenat acest model pe 45 de picturi personale, cuprinde cel mai potrivit, în viziunea mea, esența unei picturi.

Detaliile din aceste desene generate pot părea difuze, se pot observa urmele pensulei, aducând în plus trăsături realiste desenelor generate în aplicație. Spre deosebire de restul modelelor de pe aplicație, mi-a atras atenția felul în care aceste imagini par cu adevărat niște picturi realizate de artiști umani, nu pare nimic digital și oferă utilizatorului o scurtă experiență de artist.



5.4 ControlNet

ControlNet este un model de rețea neurală utilizat pentru a controla modelele Stable Diffusion în contextul generării de imagini. Folosindu-te în adiție de ControlNet, imaginile vor fi mult mai definite, pentru că adaugă o condiționare suplimentară modelelor Stable Diffusion, permitând un control mai precis asupra imaginilor generate. Această condiționare suplimentară poate lua diverse forme, cum ar fi detectarea marginilor sau detectarea poziției umane.

ControlNet este conceput pentru a îmbunătăți consistența spațială în generarea de imagini prin furnizarea de condiții de intrare suplimentare care ghidează modelul în generarea imaginilor dorite. Imaginea inițială va trece printr-o serie de procesări, astfel încât să poată fi înțeleasă mai bine.

5.4.1 Preprocesorul și Modelul ales

Preprocesorul scribble/xdog, și modelul pe care l-am ales, `control_v11p_sd15_scribble`, au în vedere că imaginea pe care o va primi Stable Diffusion este un desen. Astfel, indiferent de imaginea primită, va face background-ul negru și accentele în alb, ca mai departe să fie ușor de transformat în pictură.

Ca și exemplu, voi da o fotografie cu mine care are mai multe detalii, și vom vedea cum reușește acesta să o modifice.



Astfel, acest model este ideal pentru ce ne dorim noi să facem în cadrul aplicației, adică să transformăm un simplu desen într-o pictură, deoarece modelul va completa liniile, va modifica întreg stilul și va recreea texturile conform stilului ales.

5.5 Concluzie

Inteligenta artificială redefineste constant paradigmile artistilor și creatorilor din domeniul artei digitale, iar picturile generate prin intermediul acestor tehnologii revoluționare devin tot mai sofisticate și își găsesc un loc aparte în arta contemporană.

Consider că această aplicație reușește să pună la dispoziție posibilitatea de a crea de unul singur această artă, fără necesitatea de a cunoaște întreg procesul din spate. Astfel, imaginile create sunt rezultatul unei sinergii complexe între imaginația umană și abilitatea algoritmului de a interpreta și de a genera în conformitate cu aceasta.

Aceasta nouă eră a artei digitale aduce cu sine tehnici precum Rețelele Generative Adversariale (GANs). Am prezentat astfel modelul de difuzie în comparație cu GAN, astfel prin acest model, datele sunt transformate treptat și invers, permitând perfecționarea lor și generarea unor modele realiste. Aceasta abordare permite artiștilor digitale să experimenteze și să dezvolte imagini de înaltă calitate. Stable Diffusion în colaborare cu ControlNet reușește să demonstreze că uneori, tehnologia depășește realitatea.

Capitolul 6

Direcții de viitor

Îmbunătățirile pe care aş dori să le aduc în cadrul acestei aplicații se împart în patru categorii. Cum doresc să procedez mai departe cu această aplicație să poată fi folosită de alte persoane, plus îmbunătățirile care țin de componenta de blog, care țin de componenta de desen și în cele din urmă adaosurile pentru componenta de inteligență artificială.

Găzduirea aplicației

Primul pas pentru a putea găzdui aplicația, a fost deja implementat în proiect. Astfel, aplicația este izolată și se află într-un container, pregătită pentru a funcționa pe medii externe. Îmi doresc să pot pune la dispoziție persoanelor din jurul acest proiect, să îl îmbunătățesc și să îl pot folosi mai departe.

Un blog mai interactiv

Componenta de blog poate să devină cu siguranță mai interactivă, incluzând pe viitor posibilitatea de a conversa în cadrul aplicației. De asemenea, aş dori să includ o componentă de apreciere a postărilor, iar pe baza acesteia să compun un algoritm care să genereze postări similare cu cele apreciate.

O aplicație de desen ușor de folosit

Cât despre componenta de desen, mi-ăș dori să aduc în cadrul acesteia mai multe funcționalități, precum:

- Fiecare stroke să devină un obiect, permitând modificarea, micșorarea, mărirea sau ștergerea individuală ca un obiect de sine stătător.

- Adăugarea unei componente de blurare și estompare.
- Diversificarea tipurilor de pensule, inclusiv pensule punctate pentru a imita o pensulă reală, cu diferite intensități.
- Posibilitatea de a insera imagini de pe dispozitivul propriu.

Optiuni mai ample de modele

Inteligenta artificială o să continue să evolueze, astfel că apar zi de zi tot mai multe modele de antrenare care impresionează tot mai mult.

Deși consider că esența proiectului este pictura, ar putea fi o variantă plăcută pentru utilizatori să își aleagă ei stilul de desen. Astfel mi-aș dori să includ mai multe preprocesoare și modele din care aceștia să aleagă. Aș dori să lărgesc orizontul pentru tematica desenelor, incluzând variații de fotografii hiperrealiste și variații de stiluri joviale, precum desene animate și elemente ce au legătură cu animațiile japoneze.

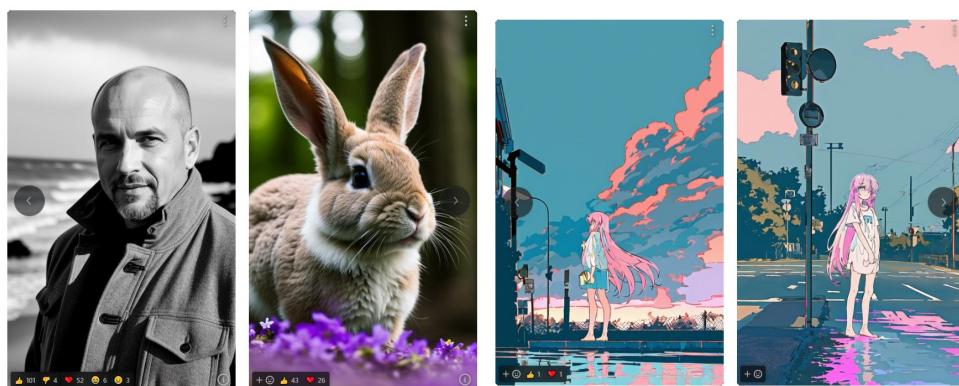


Figura 6.1: Realistic Vision V2.0 vs [LORA]Flat_Color

Concluzii

Lucrarea de față a prezentat abordarea mea asupra unei platforme interactive de desen digital, pe al său nume BlankCanva. BlankCanva este o platformă de blog, a cărei piesă principală este desenul obținut prin artă generativă.

În această lucrare am abordat tehnologiile folosite, motivația din spatele alegerii acestora prin comparații cu alte tehnologii mai populare și de încredere pentru programatori.

Pentru felul în care am dezvoltat această aplicație, am descris întreaga arhitectură din mai multe puncte de vedere. Fie că este cea server-client, diagramele C4, sau arhitectura bazei de date, lucrarea este transparentă în privința structurii ei, astfel că arhitectura server-client furnizează funcționalitatea de transmitere de informații între interfață și server, vizând astfel și tehnologiile care au susținut această arhitectură.

Sunt prezentate în detaliu și scenariile de utilizare ale aplicației. Fiecare funcționalitate de la nivelul acestelui este descrisă minuțios oferind o perspectivă amplă asupra aplicației. Am putea considera această documentație și un ghid de utilizare al ei.

Arhitectura bazei de date conține 11 tabele, acestea conținând toate informațiile importante despre utilizatori, autentificare, postări, fotografii, comentarii, bookmarks, etc.

Inteligenta artificială aduce un bonus acestui blog, fiind un subiect de actualitate pentru societate. Consider că cele două jumătăți funcționale ale proiectului, acestea fiind platforma interactivă și desenul digital, se îmbină perfect, căci ar putea aduce noi funcționalități și experiențe în mediul social media. Aceste tehnologii pot îmbunătăți interacțiunea și implicarea utilizatorilor, contribuind la evoluția și extinderea comunității de artiști digitali.

Nu în ultimul rând, am adus acestei lucrări un capitol cu implementări de ajutor pe viitor, cum ar putea fi actualizată și îmbunătățită, iar acesta este punctul de plecare pe mai departe pentru acest blog, BlankCanva.

Bibliografie

1. Introducere

- (a) López de Mántaras, Ramón. "Artificial Intelligence and the Arts: Toward Computational Creativity." In *The Next Step. Exponential Life*. Madrid: BBVA, 2016.
- (b) **Documentația oficială T3 Stack**
<https://create.t3.gg/en/introduction>
- (c) **Documentația oficială Next.js**
<https://nextjs.org/docs>
- (d) **Documentația oficială TypeScript**
<https://www.typescriptlang.org/docs/>
- (e) **Documentația oficială Tailwind CSS**
<https://tailwindcss.com/docs/installation>
- (f) **Documentația oficială tRPC**
<https://trpc.io/docs>
- (g) **Documentația oficială Prisma**
<https://www.prisma.io/docs>
- (h) **Documentația oficială NextAuth.js**
<https://next-auth.js.org/getting-started/introduction>
- (i) **Microsoft Paint**
https://ro.wikipedia.org/wiki/Microsoft_Paint
- (j) **Microsoft Paint Features**
<https://www.techwalla.com/articles/microsoft-paint-features>
- (k) **DeviantArt**
<https://www.deviantart.com/>

2. Aspecte tehnice ale tehnologiilor utilizate

(a) **Introducere T3**

<https://create.t3.gg/>

(b) **Knowing Next.js vs. React for Developers**

<https://geekflare.com/nextjs-vs-react/>

(c) Figura 2.1 : **HTML document and the associated DOM tree**

<https://www.mdpi.com/2504-4990/3/1/6>

(d) Figura 2.2 : **Tailwind CSS vs. CSS**

<https://www.programonaut.com/tailwind-css-vs-css-an-in-depth-comparison-speed-file-size-etc/>

(e) **DayJS**

<https://day.js.org/docs/en/display/format>

3. Dezvoltarea aplicației web

(a) **Adapters**

<https://trpc.io/docs/server/adapters>

(b) Figura 3.5 : **Oauth2**

<https://developers.google.com/identity/protocols/oauth2>

(c) **Inspiratie pentru diagrame**

<https://static.structurizr.com/workspace/36141/diagrams/SystemContext.png>

<https://static.structurizr.com/workspace/76748/diagrams/Components.png>

<https://static.structurizr.com/workspace/76748/diagrams/Containers.png>

<https://www.archimatetool.com/blog/2020/04/18/c4-model-architecture-viewpoint-and-archi-4-7/>

(d) **Supabase**

<https://supabase.com/docs/guides/getting-started/tutorials/with-nextjs>

(e) **Docker**

<https://dev.to/amoniacou/what-is-docker-why-is-it-important-and-necessary-for-developers-part-i-39e5>

<https://docs.docker.com/get-started/overview/>

Figura 3.11 : <https://bikramat.medium.com/docker-objects-e561f0ce3365>
<https://create.t3.gg/en/deployment/docker>

4. Componenta de desen digital

(a) **HTML Canvas**

https://www.w3schools.com/graphics/canvas_intro.asp
https://www.w3schools.com/graphics/canvas_drawing.asp

(b) **Base64**

<https://www.npmjs.com/package/base64-arraybuffer>
<https://en.wikipedia.org/wiki/Base64>

5. Inteligență artificială în arta digitală, Direcții de viitor

(a) **How to Make Computer Generated Art Using AI**

<https://www.makeuseof.com/how-to-make-computer-generated-ai-art/>

(b) **What Is AI Art? A Guide on How It Works and How to Create It.**

<https://www.makeuseof.com/how-take-computer-generated-ai-art/>

(c) **Stable Diffusion**

https://en.wikipedia.org/wiki/Stable_Diffusion

(d) **Choosing the Best Stable Diffusion Sampler: A Comprehensive Guide**

<https://neuroflash.com/blog/choosing-the-best-stable-diffusion-sampler-a-comprehensive-guide/>

(e) **Civitai**

<https://civitai.com/models/4201/realistic-vision-v20>

<https://civitai.com/models/6433/loraflatcolor>

<https://civitai.com/models/85524/scribblexdog>

<https://civitai.com/models/28068/impressionism-oil-painting>

(f) **control_v11p_sd15_scribble**

https://huggingface.co/llyasviel/control_v11p_sd15_scribble

(g) **ControlNet**

<https://stablediffusionweb.com/ControlNet>

(h) Comparison between Diffusion Models vs GANs (Generative Adversarial Networks)

<https://machinelearningknowledge.ai/comparison-between-diffusion-models-vs-gans-generative-adversarial-networks/#:text=Although%20Both%20Diffusion%20Models%20and,generate%20diverse%20real%2Dlike%20data>

(i) What Is Stable Diffusion and How Does It Work?

<https://www.vegaitglobal.com/media-center/knowledge-base/what-is-stable-diffusion-and-how-does-it-work>

(j) A Gentle Introduction to Generative Adversarial Networks (GANs)

Figurile 5.1, 5.2, 5.3 : <https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>

(k) How diffusion models work: the math from scratch

<https://theaisummer.com/diffusion-models/>

(l) Deep Convolutional Generative Adversarial Network

<https://www.tensorflow.org/tutorials/generative/dcgan>

(m) What is Prompt

<https://www.arimetrics.com/en/digital-glossary/prompt>