UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

**PROIECT DE DIPLOMĂ**

Conducător ştiinţific: Asist. Dr. Ing. Dorobanțiu Alexandru

Îndrumător: Asist. Dr. Ing. Dorobanțiu Alexandru

**Absolvent**

Banea Marcus-Andrei

**Specializarea**

Calculatoare

- Sibiu, 2023 -

UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

**Concurs de boți implementat**

**in cadrul unui joc de carti**

Conducător ştiinţific: Asist. Dr. Ing. Dorobanțiu Alexandru

Îndrumător: Asist. Dr. Ing. Dorobanțiu Alexandru

**Absolvent**

Banea Marcus-Andrei

**Specializarea**

Calculatoare

- Sibiu, 2023 -

Cuprins

[Cuprins 3](#_Toc136713373)

[1. Prezentarea temei (2-4 pagini): 4](#_Toc136713374)

[1.1. Introducere 4](#_Toc136713375)

[1.2. Descrierea temei 4](#_Toc136713379)

[1.3. Scop si obiective 4](#_Toc136713382)

[1.3.1. Obiective principale 5](#_Toc136713384)

[1.3.1.1. Obtinerea setului de carti 5](#_Toc136713385)

[1.3.1.2. Realizarea interfetei jocului 7](#_Toc136713403)

[1.3.1.3. Implementarea mecanicii jocului 8](#_Toc136713412)

[1.3.2. Obiective secundare 8](#_Toc136713414)

[1.4. Tehnologii utilizate 8](#_Toc136713421)

[1.4.1. Biblioteci 9](#_Toc136713428)

[1.5. Cazuri principale de utilizare a aplicatiei 9](#_Toc136713431)

[1.6. Organizarea lucrarii 9](#_Toc136713433)

[2. Teorie (25-30 pagini) 9](#_Toc136713434)

[2.1. Source Control 9](#_Toc136713435)

[2.1.1. GIT 9](#_Toc136713436)

[2.1.2. SmartGit 9](#_Toc136713437)

[2.2. Java 9](#_Toc136713438)

[2.3. Javascript 9](#_Toc136713439)

[2.4. Vue.js 9](#_Toc136713440)

[2.5. Pinia 9](#_Toc136713441)

[2.6. Xstate 10](#_Toc136713442)

[2.7. MongoDB 10](#_Toc136713443)

[2.8. IDEs 10](#_Toc136713444)

[2.9. Wireframes 10](#_Toc136713445)

[2.10. Justinmind Mockups 10](#_Toc136713446)

[3. Descrierea formala a aplicatiei (5-15 pagini) 10](#_Toc136713447)

[3.1. Actori 10](#_Toc136713448)

[3.2. Use 10](#_Toc136713449)

[3.2.1. Admin Use-case 10](#_Toc136713450)

[3.2.2. Player Use-case 10](#_Toc136713451)

[3.2.3. System Use-case 10](#_Toc136713452)

[3.3. Wireframes 10](#_Toc136713453)

[3.4. Mockups 10](#_Toc136713454)

[3.5. Arhitectura 10](#_Toc136713455)

[4. Detalii de implementare (25-30 pagini) 10](#_Toc136713456)

[Descriu use-case 10](#_Toc136713457)

[5. Concluzii si dezvoltari ulterioare (1-2 pagini) 11](#_Toc136713458)

[5.1. Concluzii 11](#_Toc136713459)

[5.2. Dezvoltare ulterioare 11](#_Toc136713460)

[6. Bibliografie 11](#_Toc136713461)

1. Prezentarea temei (2-4 pagini):

1.1. Introducere

Chemarea mea catre acest proiect isi are originile in unele dintre cele mai frumoase perioade ale vietii: copilaria, si perioada studentiei. Jocul de carti Duel Masters, unul dintre obiectele principale ale acestui proiect, a facut parte din copilaria mea, fiind captivat atat de interactiunea pe care o aveam cu ceilalti prieteni cu care jucam, cat si de complexitatea acestuia.

Anii au trecut, copilaria a ramas in urma, iar ideea acestui joc a ramas pentru mine doar o amintire. In schimb, o alta pasiune a aparut: programarea. Liceul m-a introdus in aceasta lume, iar studentia mi-a aratat ca singurul obstacol din acest domeniu este imaginatia.

Astfel, avand oportunitatea de alege o tema de proiect pentru incheierea studiilor universitare, am decis sa utilizez cunostiintele dobandite in ultimii ani si sa combin cele 2 elemente care au marcat cele 2 perioade ale vietii mele.

1.2. Descrierea temei

Dupa cum se precizeaza si in titlul temei alese, aplicatia dezvoltata va presupune realizarea unui mediu de desfasurare a unui meci de Duel Masters, intre boti. Acesti boti vor inlocui jucatorul real, si vor lua decizii in functie de contextul si starea meciului. Calitatea deciziilor va fi diversa, intru-cat jocul este unul complex.

Totodata, aplicatia va permite desfasurarea de meciuri intre jucatori reali (useri) si boti. Nivelul de dificultate al meciului va fi reprezentat de catre algoritmul din spatele bot-ului. Despre metodologia de luare a unei decizii se va vorbi, mai tarziu, in capitolul ...

1.3. Scop si obiective

Dezvoltarea aplicatiei a urmarit cateva obiective principale, prioritare, care reprezentau baza temei. Pe langa acestea, vom prezenta si cate obiective secundare, dezvoltate pe baza celor principale, ce aduc noi functionalitati si imbogatesc placut aplicatia.

1.3.1. Obiective principale

1.3.1.1. Obtinerea setului de carti

Dupa cum am precizat anterior, jocul Duel Masters reprezinta un joc de carti. Fiecare carte are utilizari diferite, determinate de diferiti factori precum: starea curenta a meciului, mutarile facute de cei 2 jucatori in tururile anterioare ale meciului, etc.



Figura 1.0: Carte de tip Creature

Astfel, avand in considerare timpul disponibil dezvoltarii acestui proiect, cat si dorinta autorului de a realiza o aplicatie functionala, care sa prezinte cat mai multe dintre elementele acestui joc, a fost necesara trierea setului complet de carti ale jocului.

Setul contine 3 tipuri principale de carti:

1. Creature (creaturi) : carti pe care cei 2 jucatori le plaseaza pe terenul de joc si care participa la dueluri intre ele, fiind elementul principal al jocului. O astfel de carte este prezentata in figura 1.1.
2. Spell: carti cu one-time effect; cu alte cuvinte, aceste carti sunt utilizate pentru a realiza o actiune (abilitate) in momentul utilizarii, dupa care sunt eliminate din joc
3. Evolution (evolutii): carti similare cu cele de tip Creature, diferenta reprezentand-o necesitatea de a plasa o astfel de carte deasupra unei carti de tip Creature, realizand, dupa cum ii spune si numele, o evolutie a cartii de baza.



Figura 1.1: Carte de tip Spell

Cartile de tip Creatura reprezinta majoritatea cartilor din setul complet de carti, iar din acest motiv s-a ales utilizarea, pentru dezvoltarii primei versiuni a aplicatiei, doar acest tip de carti.



Figura 1.2: Carte de tip Evolution

Alti factori care au contribuit la trierea setului de carti:

* Limitarea la doar cateva tipuri speciale de Creatura (eliminarea celor de tip Cross Gear, Wave Striker sau Accelerator)
* Imaginile cartilor propriu-zise: intru-cat jocul isi are originile in Japonia, exista carti pentru care nu s-au putut obtine imagini cu textul inscris pe acestea, in limba engleza.

Dintr-un total de aproximativ 1400 de carti, s-a obtinut un set de 433 de carti de tip Creatura ce vor fi utilizate in prima faza a dezvoltarii aplicatiei.

1.3.1.2. Realizarea interfetei jocului

Pentru desfasurarea unui meci, fie el intre 2 boti, 1 bot si un jucator real (user) sau intre 2 jucatori (planificat pentru dezvoltari ulterioare), este necesara crearea unei interfete vizuale, in care sunt dispuse cele 2 table de joc, corespunzatoare fiecarui jucator.

Stuctura unei table de joc este urmatoarea:

* Battle Zone: zona de batalie, unde sunt plasate cartile de tip Creatura ce se vor duela cu cele ale jucatorului advers.
* Mana Zone: mana reprezinta resursa necesara pentru a putea plasa in zona de batalie o carte. Fiecare carte va consuma o anumita cantitate de mana, aceasta valoare fiind inscriptionata pe carte. Astfel, fiecare carte plasata in zona de mana va creste capacitatea jucatorului de a plasa carti in zona de batalie.
* Hand: cartile din mana jucatorului. Acestea pot fi plasate in una din cele 2 zone prezentate anterior.
* Graveyard: zona cartilor distruse. In urma duelurilor sau executiei abilitatiilor, cartile ce vor fi distruse vor fi plasate in aceasta zona.
* Deck: pachetul de carti. In fiecare tura, jucatorul poate lua o carte din acest pachet, fapt ce asigura continuitatea meciului.
* Shields: scuturile jucatorului. Acestea reprezinta punctele de viata ale jucatorului. Jucatorul care ramane, in timpul meciului, fara nici un scut, va pierde meciul.

1.3.1.3. Implementarea mecanicii jocului

Jocul prezinta un set de reguli si mecanici care se aplica in diferite momente ale meciului. Spre exemplu: in fiecare tura, un jucator poate lua o singura carte din pachet. Aceste mecanici vor fi implementate si integrate atat pentru conceptul de meci bot vs bot, cat si in celelalte variante.

1.3.2. Obiective secundare

Obiectivul principal este, in mod evident, crearea unui mediu prin care poate avea loc un meci de Duel Masters. Realizarea acestui obiectiv necesita indeplinirea altor cateva obiective, dintre care enumeram:

* Optiunea de a crea mai multi useri (playeri)
* Posibilitatea unui user de a isi gestiona colectia de carti:
  + Vizualizarea colectiei
  + Renuntarea la cartile nedorite
  + Imbogatirea colectiei cu alte carti, prin cupararea de pachete de carti sau schimburi de carti cu alti jucatori

1.4. Tehnologii utilizate

Arhitectura aplicatiei prezinta 3 componente principale: serverul, clientul si baza de date.

Tehnologiile utilizate pentru gestionarea celor 3 sunt urmatoarele:

* Server: Limbajul de programare JAVA; Mediul de programare Eclipse (varianta SpringToolSuite4)
* Client: Limbajul de programare JavaScript; Frameworkul Vue.js; Mediul de programare Visual Studio Code
* Baza de date: baza de date document-oriented MongoDB; software-ul utilizat pentru administrarea bazei de date: MongoExpress

Totodata, pentru testarea request-urilor venite dinspre client spre server, am utilizat API-ul Postman.

1.4.1. Biblioteci

Impreuna cu framework-ul Vue.js, am utilizat librariile Pinia si Xstate.

Pinia reprezinta o librarie utila pentru managementul starilor din ecosistemul Vue. La randul ei, Xstate reprezinta o librarie utila pentru implementarea conceptului de finite state machine (aparat cu stari finite). Cele 2 s-au dovedit a fi cruciale in procesul de dezvoltare a aplicatiei, ajutand la o gestiune eficienta si fluida a informatiei cu care lucreaza aplicatia.

1.5. Cazuri principale de utilizare a aplicatiei

Aplicatia poate fi observata din 2 perspective. Partea de concurs de boti se adreseaza unui public mai rezervat, care urmareste dezvoltarea si implementarea unor algoritmi de joc asupra unui subiect, care in cazul acesta este jocul Duel Masters. Totodata, jocul propriu-zis este adresat in special copiilor, insa complexitatea acestuia poate crea interes si din partea unor persoane mai mature.

1.6. Organizarea lucrarii

2. Teorie (25-30 pagini)

2.1. Source Control

Precum orice proces de dezvoltare a unei aplicatii, nu vor exista niciodata doar 2 etape: faza initiala, in care aplicatia este complet nefunctionala, si faza finala, in care aplicatia prezinta toate functionalitatiile dorite de proiectant. Procesul va fi marcat de o succesiune de pasi ce vor fi urmati pentru a realiza produsul final.

Fara a intra in detalii specifice acestui proiect, enumeram cateva situatii care prezinta mai clar pasii amintiti anterior, cat si impedimentele provocate de acestia:

* Implementarea unei functionalitati specificate de proiectant
* Aparitia unor schimbari de proiectare, ce necesita modificarea unei functionalitati deja implementate
* Rezolvarea unor bug-uri aparute in urma implementarii

Se observa clar necesitatea pastrarii unei istorii a versiuniilor aplicatiei, pe parcursul dezvoltarii. Pentru a realiza asta, s-a decis utilizarea unui sistem de control al versiunilor unui proiect: GIT.

2.1.1. GIT

Git reprezinta unul dintre cele mai utilizate sisteme de control al versiunilor. Acesta a fost creat de catre Linus Torvalds, creatorul Linux, in anul 2005.

Este folosit pentru a salva si urmarii modificarile ce sunt efecutate asupra fisierelor unui proiect. Astfel, dezvoltatorul proiectului poate urmarii atat stadiul actual al proiectului, cat si starea acestuia la diferite momente de timp.

Aceste proprietati il transforma in tool-ul esential colaborarii in echipele de dezvoltare al unui soft. Mai multi developeri pot lucra simultan pe acelasi proiect si isi pot share-ui progresul, functionalitatile Git-ului permitand aceasta tranzitie eficienta si fara conflicte. Despre acestea se va detalia in cele ce urmeaza.



Figura 2.1.1. Salvarea datelor sub forma de snapshots

2.1.1.1. Functionalitatiile Git-ului

Elementul central in cadrul utilizarii Git-ului este bine cunoscutul Repository (sau, pe scurt, repo). Acesta reprezinta, practic, un spatiu de stocare in care se afla atat fisierele, cat si istoricul modificarilor (commit-urile).

Pentru a crea un repository, trebuie ales un folder de catre utilizator, si apoi anuntat Git-ul, printr-o

2.1.1.2. Utilizarea Git-ului

Git-ul poate fi utilizat prin diferite metode. Utilizarea command-line-ului reprezinta metoda clasica, insa pot fi utilizate si alte interfete grafice (GUI) precum SmartGit sau SourceTree.

Diferenta dintre cele 2 metode consta in facilitatile pe care le ofera cea din urma: spre deosebire de linia de comanda, unde este necesar a fi cunoscute comenzile propriu-zise, un client GUI pentru Git simplifica managementul Repository-ului.

2.1.1.3. SmartGit

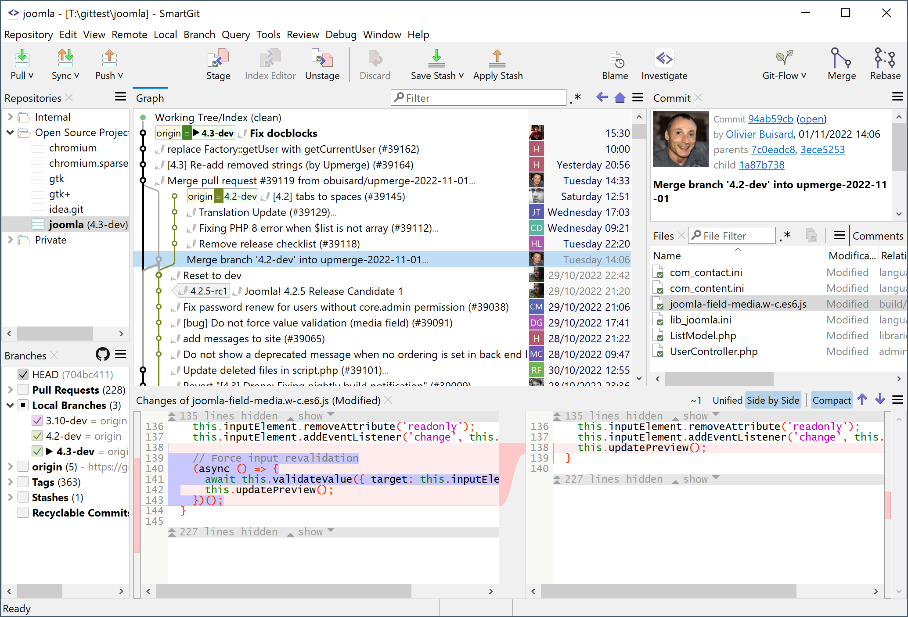


Figura 2.1.2. Interfata grafica a clientului SmartGit

2.2. Java

2.3. Javascript

2.4. Vue.js

2.5. Pinia

2.6. Xstate

2.7. MongoDB

2.8. IDEs

2.9. Wireframes

2.10. Justinmind Mockups

3. Descrierea formala a aplicatiei (5-15 pagini)

3.1. Actori

3.2. Use

3.2.1. Admin Use-case

3.2.2. Player Use-case

3.2.3. System Use-case

3.3. Wireframes

3.4. Mockups

3.5. Arhitectura

4. Detalii de implementare (25-30 pagini)

Descriu use-case

5. Concluzii si dezvoltari ulterioare (1-2 pagini)

5.1. Concluzii

5.2. Dezvoltare ulterioare

6. Bibliografie