

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAŞI
FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

Joc de cărți generalizat

propusă de

Mădălin-Marian Săvoaia

Sesiunea: *Iulie, 2018*

Coordonator științific

Lector, dr. Nicolae-Cosmin Vârlan

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

Joc de cărți generalizat

Mădălin-Marian Săvoaia

Sesiunea: Iulie, 2018

Coordonator științific
Lector, dr. Nicolae-Cosmin Vârlan

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele _____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul(a)

domiciliul în

născut(ă) la data de, identificat prin CNP,
absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de
..... specializarea, promoția, declar pe propria răspundere, cunoscând consecințele falsului în
declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr.
1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul:

elaborată sub îndrumarea dl. / d-na

....., pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență,

de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi,

Semnătură student

DECLARAȚIE DE CONSUMĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Joc de cărți generalizat*”, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent *Mădălin-Marian
Săvoaia*

(semnătura în original)

Table of Contents

1	<u>INTRODUCERE</u>	8
1.1	MOTIVĂIE.....	8
1.2	CONTEXT.....	8
1.3	CERINȚE FUNCȚIONALE.....	9
1.3.1	CERINȚE FUNCȚIONALE LA NIVEL DE DEZVOLTATOR.....	9
1.3.2	CERINȚE FUNCȚIONALE LA NIVEL DE UTILIZATOR.....	9
2	<u>CONTRIBUTIE</u>	10
2.1	LIMBAJ DESCRIERE MOD DE JOC	10
2.2	CONFIGURATOR MOD DE JOC	11
2.3	SERVER.....	11
2.4	CLIENT.....	11
3	<u>PROIECTARE</u>	12
3.1	ARHITECTURA SOLUȚIEI.....	12
3.2	MODELAREA DATELOR.....	12
3.2.1	MODELAREA DATELOR LA NIVEL DE CONFIGURATOR	12
3.2.2	MODELAREA DATELOR LA NIVEL DE SERVER	13
3.2.3	MODELAREA DATELOR LA NIVEL DE CLIENT	13
3.3	PROTOCOALE DE COMUNICARE CLIENT-SERVER	14
3.4	INTERFAȚA CU UTILIZATORUL.....	16
4	<u>IMPLEMENTARE</u>	19
4.1	ABORDARE TEHNICĂ.....	19
4.1.1	CONFIGURATOR MOD DE JOC	19
4.1.2	SERVER	19
4.1.3	CLIENT	19
4.2	IMPLEMENTARE CONFIGURATOR.....	19
4.3	IMPLEMENTARE SERVER.....	22
4.3.1	IMPLEMENTARE INTERPRETOR	23
4.3.2	IMPLEMENTARE MOD DE JOC	26
4.3.3	IMPLEMENTARE SISTEM CAMERE.....	29
4.4	IMPLEMENTARE CLIENT.....	30
4.4.1	CONFIGURARE ȘI ACTUALIZARE DEPENDENȚE EXTERNE	30
4.4.2	IMPLEMENTARE INTERFAȚĂ GRAFICĂ.....	31
4.4.3	IMPLEMENTARE COMUNICAȚIE CU SERVERUL	31
5	<u>MANUAL DE UTILIZARE</u>	33
5.1	MANUAL DE UTILIZARE CONFIGURATOR	33
5.1.1	GENERARE MOD DE JOC	33

5.2 MANUAL DE UTILIZARE SERVER.....	35
5.2.1 EXEMPLIFICARE ADĂUGARE MOD DE JOC.....	35
5.2.2 EXEMPLIFICARE CONFIGURARE SERVER	36
5.2.3 EXEMPLIFICARE RULARE SERVER.....	36
5.3 MANUAL DE UTILIZARE CLIENT	36
5.3.1 EXEMPLIFICARE CONFIGURARE CLIENT	36
5.3.2 EXEMPLIFICARE RULARE CLIENT.....	36
5.3.3 EXEMPLIFICARE ÎNCEPERE JOC	36
5.3.4 EXEMPLIFICARE COMENZI JOC.....	37
5.3.5 EXEMPLIFICARE ÎNCHEIERE JOC	37
6 CONCLUZII	39
7 ANEXE	40
8 BIBLIOGRAFIE	41

1 Introducere

În această lucrare propun dezvoltarea unui sistem ce urmărește să faciliteze dezvoltarea jocurilor de cărți generalizate, astfel încât dezvoltatorului să îi revină responsabilitatea de a genera un mod de joc după regulile dorite și de a implementa un client pe platforma dorită sau de a rafina actualul client specific platformei iOS.

În continuare am să vorbesc despre motivație, context, cerințe funcționale și abordare tehnică.

1.1 Motivație

Sesizând nevoia pieței actuale de a dezvolta jocuri rapid în aceste vremuri, am decis să construiesc acest sistem cu ajutorul căruia regulile jocului de cărți se pot transpune în practică folosind un configurator. Astfel că nu este necesară cunoașterea în detaliu al mecanismului din spate sau al limbajului în care este scris modul de joc. Acest fapt face ca modul de joc să fie construit, modificat și înțeles ușor chiar și de către cineva care nu are cunoștințe foarte avansate în domeniu.

1.2 Context

În ultima perioadă consumul de jocuri video a cunoscut o expansiune semnificativă, în special în ultimul timp pe dispozitivele mobile. Astfel că jocuri video de toate categoriile, pentru toate vârstele și pentru toate platformele apar în fiecare zi, iar dezvoltatorii sunt puși în situația în care sunt nevoiți să producă jocuri rapid, eficient și simplu, astfel încât să își mulțumească clienții.

În acest context sistemul la care m-am gândit se adaptează noilor cerințe ale pieței și poate să acopere o parte din această cerere pe parte de jocuri de cărți generalizate, bazate pe niște moduri de joc.

În continuare am să dau două exemple de proiecte care au abordat un subiect similar. Primul dintre ele este **Dekr**¹, un proiect susținut de comunitatea GitHub, care are ca scop construirea unui sistem reutilizabil și extensibil pentru a crea jocuri de cărți. Acest sistem oferă suport pentru dezvoltarea de jocuri precum Solitaire, Hearts și Dominion, dar din păcate acest proiect nu a mai fost actualizat din 2014.

Al doilea exemplu de proiect care a abordat o temă similară este **CCG Toolkit**², un framework pentru Unreal Engine dezvoltat de către Aaron Scott, cu ajutorul căruia dezvoltatorii pot să construiască jocuri de cărți utilizând blueprint. Acesta are ca avantaj faptul că dezvoltatorul pune la dispoziție o documentație bună și oferă suport pe un forum dedicat.

¹ <https://github.com/courageousillumination/dekr>

² <https://www.unrealengine.com/marketplace/ccg-toolkit>

1.3 Cerințe funcționale

În cele ce urmează am să prezint cerințele funcționale ale acestui proiect, care se împart în două categorii, și anume partea care face referire la dezvoltator și partea care face referire la utilizator.

1.3.1 Cerințe funcționale la nivel de dezvoltator

Fiind un sistem cu ajutorul căruia dezvoltatorii construiesc jocuri de cărți generalizate multiplayer, avem niște cerințe funcționale referitoare la dezvoltator, iar acestea ar fi următoarele:

- Construire mod de joc în editor text, utilizând drept exemplu două dintre modurile de joc construite în faza de dezvoltare
- Construire mod de joc cu ajutorul unui configurator web pentru a eficientiza procesul de dezvoltare
- Modalitate simplă de configurare adresă și port server
- Modalitate simplă de configurare adresă și port client
- Rulare server cu ajutorul Python 2.7³

1.3.2 Cerințe funcționale la nivel de utilizator

Când vine vorba despre utilizatorul final, cerințele funcționale ar fi următoarele:

- Distribuire aplicație iOS prin intermediul AppStore⁴, TestFlight⁵ sau Diawi⁶
- Accesare funcționalitate joc prin intermediul butonului play din meniul principal
- Luarea unei cărți din pachet
- Anularea acțiunii unei cărți, date de jucătorul anterior
- Selectarea și lăsarea cărților selectate pe masă
- Câștigarea jocului de utilizatorul care a atins primul un anume prag de cărți

³ <https://www.python.org>

⁴ AppStore este piața de desfacere a produse software pentru dispozitivele Apple

⁵ TestFlight este un sistem de la Apple care facilitează distribuirea aplicațiilor pentru testare

⁶ Diawi este un sistem care ajută la distribuirea aplicațiilor în afara AppStore

2 Contribuție

Acest proiect are ca scop dezvoltarea unui sistem cu ajutorul căruia dezvoltatorii să creeze diferite jocuri de cărți multiplayer într-un timp relativ scurt și fără să aibă nevoie de cunoștințe foarte avansate de programare. Astfel că proiectul este împărțit în patru componente principale, limbaj descriere mod de joc, configurator mod de joc, server și client. Pe parcursul acestei lucrări vom urmări dezvoltarea componentelor menționate anterior, cu detalii despre implementare și ilustrațiile aferente.

2.1 Limbaj descriere mod de joc

În cadrul acestui proiect m-am gândit să construiesc un limbaj cu ajutorul căuia să pot descrie un joc de cărți. Ideea a pornit de la jocul de cărți macao și dorința de a putea modifica într-un fel sau altul modul de funcționare a jocului cu ajutorul limbajului natural.

Primul pas a fost să pot configura numele jocului, iar pentru asta am introdus cheia GAME_MODE_NAME. Mai apoi m-am gândit că este relevant să pot specifica numărul de cărți de început pentru fiecare jucător, iar pentru asta am introdus cheia NUMBER_OF_CARDS_PER_PLAYER.

Văzând că ideea de a construi un limbaj pentru a configura un joc de cărți funcționează, am introdus în mod similar chei pentru numarul maximum de jucători, numărul minimum de jucători, numărul de cărți necesar pentru a câștiga, numărul maximum de cărți într-o singură tură, cât și pentru specificarea ordinii inițiale ale jucătorilor, aleatorie sau nu.

În cele din urmă, limbajul putea descrie câteva caracteristici simple ale unui joc de cărți, dar încă nu era de ajuns de puternic pentru a descrie un joc în sine, aşa că am continuat dezvoltarea acestuia. Principala problema la acel moment era că nu aveam încă o modalitate de a specifica ce fel de cărți se pot pune peste cartea curentă, iar acest lucru l-am considerat căt se poate de necesar în contextul în care voiam în cele din urmă să pot descrie regulile unui joc.

Începând să studiez problema relatată anterior, mi-am dat seama că mă interesează să pot să atribui niște acțiuni fiecărei cărți în parte, aşa că simpla introducere a unor atrbute simple nu mă mai ajuta în acest caz. Astfel am introdus cheia CARDS_ACTIONS, care conține toate cărțile de joc, numerotate de la 1 la 14, iar în cadrul fiecărei cărți între delimitatorii BEGIN și END se pot introduce chei precum NEXT_CARDS_ACCEPTED - care poate lua ca valoare o listă de cărți sau pur și simplu ALL, ACCEPT JUST SAME_SYMBOL - care poate lua ca valoare TRUE sau FALSE, ADD_NEXT_PLAYER_CARDS - care poate lua ca valoare un număr, REMOVE_ADD_NEXT_PLAYER_CARDS - care are ca valoare de asemenea TRUE sau FALSE și REMOVE_NEXT_PLAYER_TURNS - care ia ca valoare un număr.

În acest context în care limbajul devine mai expresiv, iar cărțile încep să aibă anumite roluri în joc, rămâne totuși un caz pe care încă nu îl acoperă limbajul, și anume nu poate descrie dacă unele

cărți se pot da sau nu în grup cu altele. Astfel că m-am gândit să introduc cheia ACCEPTED_CARDS_GROUPS_IN_ONE_TURN, având ca delimitatori BEGIN și END, iar în interiorul acestora, pe fiecare linie în parte se regăsesc liste de cărți care se pot da într-o singură tură, iar în acest fel am reușit să cresc gradul de expresivitate.

După ce am parcurs pașii enumerați anterior, rezultatul s-a dovedit a fi un limbaj destul de bun pentru a descrie un joc de cărți precum macao, însă pe același principiu se mai pot adăuga anumite chei simple, cât și acțiuni pentru cărțile curente astfel încât să putem descrie jocuri mai complexe.

2.2 Configurator mod de joc

Având în vedere că este destul de costisitor din punct de vedere al timpului să te documentezi și să scrii într-un limbaj pe care nu îl cunoști, am decis să creez un configurator prin care dezvoltatorul completează un formular care îi generează la final modul de joc. Prin urmare nu sunt necesare cunoștințe foarte specifice de limbaj.

2.3 Server

Acesta se bazează pe arhitectura server-client, utilizează *socketi* și implementează un protocol de comunicare. În spate acesta funcționează ca un sistem stare-tranziție, în care serverul prezintă starea tuturor aplicațiilor de tip client, iar acestea din urmă răspund cu una din opțiunile aferente dacă este cazul. În cele din urmă, după schimbarea stării sistemului, aceasta este comunicată aplicațiilor de tip client.

2.4 Client

Când vine vorba de contribuții la nivelul aplicației client, aş putea aminti interfața grafică realizată cu ajutorul UIKit⁷, mecanismul de conectare și management al transmisiei de informații cu serverul realizat de asemenea cu SocketSwift, după cum am precizat și în capitolul precedent și mecanismul prin care se face managementul interacțiunii cu interfața grafică.

⁷ <https://developer.apple.com/documentation/uikit>

3 Proiectare

3.1 Arhitectura soluției

Pentru dezvoltarea proiectului am adoptat modelul client-server. Din moment ce modul de joc a fost scris, serverul configurat și pornit, aplicația client se poate conecta la server prin *socketi*, iar jocul începe.

O diagramă a arhitecturii se poate vedea în Figura 1.

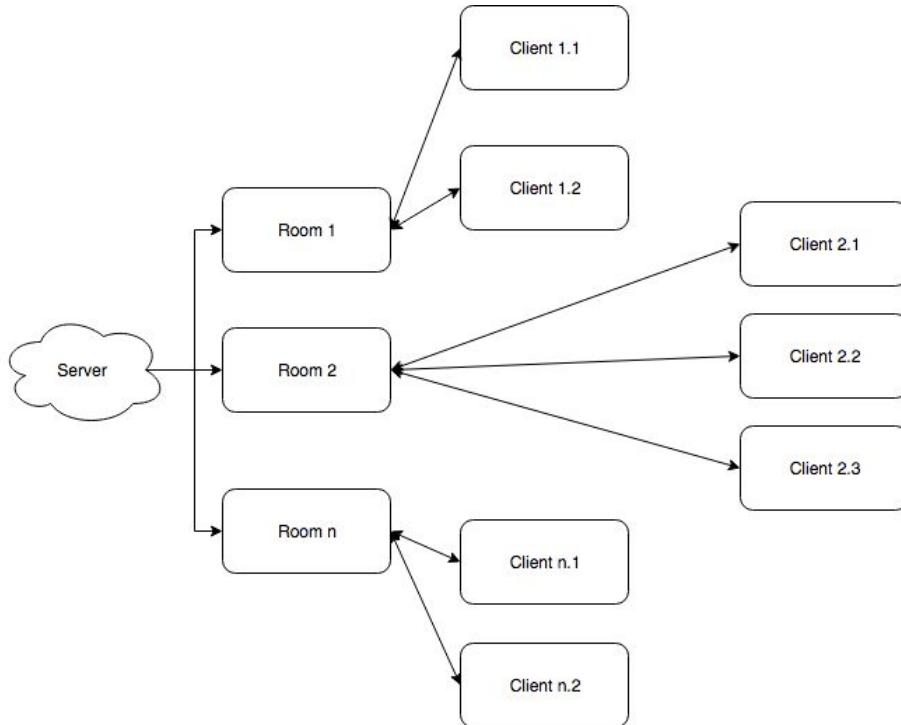


Figura 1 - Arhitectura sistemului

3.2 Modelarea datelor

Acest proiect având trei componente, configurator, server și client, modelarea datelor este prezentă în fiecare dintre componente. Astfel, această secțiune va fi împărțită în modelarea datelor la nivel de configurator - unde vom analiza datele implicate în procesul de creare a regulilor de joc, modelarea datelor la nivel de server - unde vom analiza datele implicate din momentul pornirii și până în momentul închetării rulării serverului și modelarea datelor la nivel de client, unde de asemenea vom analiza datele implicate în procesul de rulare al aplicației mobile.

3.2.1 Modelarea datelor la nivel de configurator

Pe parte de configurator, datele sunt introduse de către dezvoltator prin intermediul formularului web, care cu ajutorul unui program în PHP generează un fișier text scris într-un limbaj

bazat pe token - valoare. Acest fișier reprezintă setul de reguli pe care se va baza ulterior serverul de joc.

3.2.2 Modelarea datelor la nivel de server

Când vine vorba de server, acesta lucrează cu fișierul de reguli generat anterior prin intermediul configuratorului, pe care îl gestionează cu ajutorul unui interpretor. Astfel că putem să inițializăm obiectul de tip CustomGame cu datele deținute, obiect care reprezintă de fapt modul de joc folosit la acea rulare a serverului.

3.2.3 Modelarea datelor la nivel de client

Pe parte de client lucrurile stau puțin diferit în sensul că avem două tipuri de date cu care aplicația ia contact, și anume date statice și date dinamice să le spunem.

Datele statice sunt reprezentate de colecții de imagini grafice care compun interfața cu utilizatorul, un bun exemplu este cel din Figura 2.

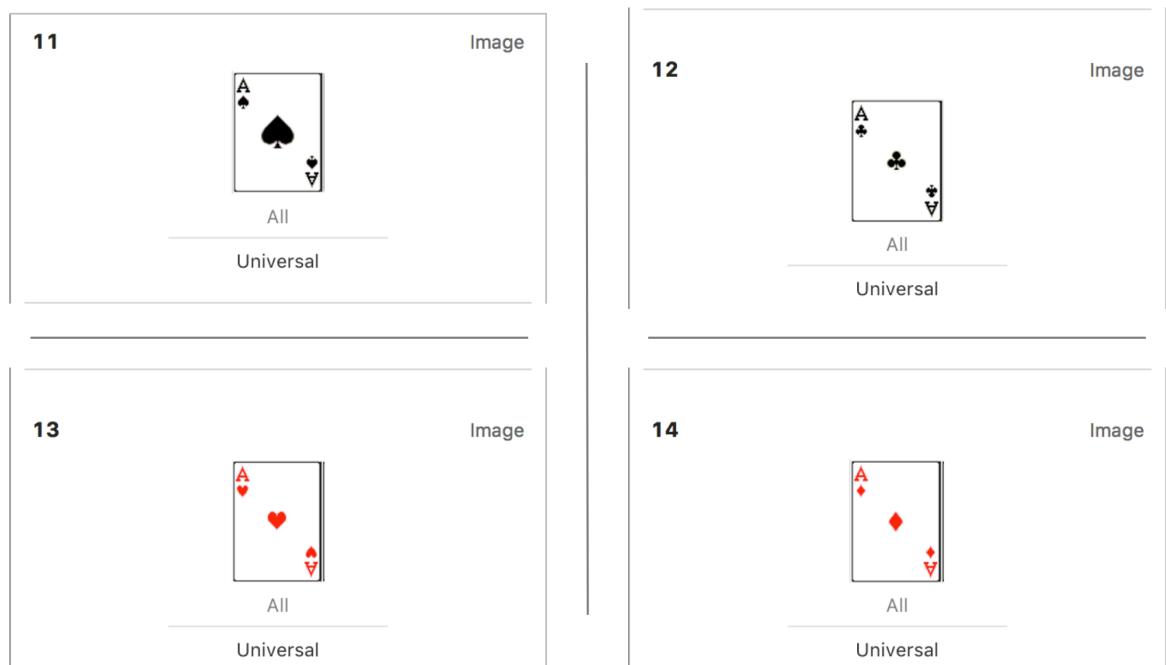


Figura 2 - Colecție de imagini grafice. Sursa:<http://www.milefoot.com/math/discrete/counting/cardfreq.htm>

Datele dinamice se referă la stările serverului din timpul jocului. Acestea se pot schimba în funcție de interacțiunea clienților, interacțiune care este limitată la niște acțiuni cum ar fi luarea unei cărți din pachet, punerea unei cărți valide jos, punerea unui grup de cărți valide jos.

În Figura 3 este ilustrată o stare serverului la un anumit moment, aşa cum este disponibilă pentru aplicația client. Din aceasta putem observa că serverul trimite starea sa către client în format JSON, unde cheile au următoarele semnificații:

- isYourTurn - este o valoare booleană, care specifică dacă este rândul jucătorului în cauză sau nu.
- code - este o variabilă care transmite anumite informații, ca de exemplu dacă jocul este început, în derulare sau încheiat.
- cardsToTake - este o colecție care conține cărțile pe care trebuie să le ia jucătorul curent.
- currentCard - reprezintă ultima carte de pe masă.
- canGetNewCard - este o valoare booleană, care specifică dacă jucătorul curent poate să tragă o carte de jos sau nu.
- validCards - este o colecție care conține cărțile pe care jucătorul poate să le pună jos.
- waitTimes - este o variabilă de tip integer care specifică câte runde trebuie să aștepte jucătorul curent, în caz că a fost blocat de alt jucător.
- validGroupsOfCards - este o colecție de colecții de cărți care specifică ce grupuri de cărți pot fi lăsate jos de către jucător dintr-o singură mână.
- cards - este o colecție de cărți care conține toate cărțile jucătorului la cel moment.



```
Last login: Thu May 10 17:35:45 on ttys002
[EN610518:~ madalin.savoaia$ telnet 127.0.0.1 8889
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^']'.
{"isYourTurn": false, "code": "2", "cardsToTake": [], "currentCard": "82", "canGetNewCard": false, "validCards": ["122", "42"], "waitTimes": 0, "validGroupsOfCards": [], "cards": ["31", "14", "23", "122", "42"]}]
```

Figura 3 - Stare a serverului la un anumit moment

3.3 Protocole de comunicare client-server

Comunicarea dintre client și server este realizată cu ajutorul *socketilor* și a protocolului de comunicare TCP/IP, care permit o comunicare bidirectională între entitățile care doresc să comunice.

La nivel de client am folosit SwiftSocket, o librărie distribuită cu ajutorul la CocoaPods, librărie care facilitează lucrul cu *socketi* la nivel înalt.

În Figura 4 putem vedea funcția de conectare la server, în cadrul căreia dacă:

- conectarea se efectuează cu succes, aplicația continuă să aștepte mesaje de la server și să răspundă acestuia, pe un fir separat de cel principal
- conectarea eșuează, aplicația afișează un mesaj de eroare către utilizator

```
func connectToServer() {
    self.client = TCPClient(address: host, port: Int32(port))

    if self.connected == false {
        guard let client = self.client else { return }

        switch client.connect(timeout: 10) {
        case .success:
            DispatchQueue.global(qos: .default).async {
                while(true) {
                    if let response = self.readResponse(from: self.client!) {
                        DispatchQueue.main.async(execute: {
                            self.layer.isHidden = true
                            self.handleServerResponse(response: response)
                        })
                    }
                }
            }
        case .failure:
            DispatchQueue.main.async(execute: {
                self.showMessage(title: "Attention", message: "Game server connection failed !",
                                dismiss: true)
            })
        }
    }
}
```

Figura 4 - Ilustrare modalitate conectare la server

La nivel de server am folosit modulul Socket din librăria standard Python, care ne facilitează următoarele acțiuni:

- Creare *socket*, legare cu adresă și port și așteptare client. (Figura 5)
- Acceptare conexiune cu clientul. (Figura 6)

```
20 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21 print ('Socket created')
22
23 try:
24     s.bind((HOST, PORT))
25 except socket.error as msg:
26     print ('Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1])
27     sys.exit()
28
29 print ('Socket bind complete')
30
31 #Start listening on socket
32 s.listen(10)
33 print ('Socket now listening')
~
```

Figura 5 - Ilustrare folosire Socket

```

270 while True:
271     print ("Server in asteptare ...")
272     connection, address = s.accept()
273     threading.Thread(target=gameHandler, args=(connection,)).start()
274     print ("While True end")
275

```

Figura 6 - Ilustrare așteptare client

3.4 Interfața cu utilizatorul

Când vine vorba de interfața cu utilizatorul pe dispozitivele mobile, lucrurile se complică într-o anumită măsură, și acest fapt nu este datorat neapărat implementării acestei interfețe. Acest fapt se datorează proiectării acestei interfețe grafice astfel încât să fie simplă, clară, eficientă și să ofere utilizatorului o experiență bună în folosință.

În cazul ideal aceasta ar trebui să fie proiectată de o persoană specializată, însă în prezență lucrare scopul este de a valorifica acest server de cărți generalizat și de a oferi dezvoltatorilor un punct de plecare în dezvoltarea jocului pe parte de client.

Acestea fiind spuse, interfața cu utilizatorul conține două ecrane.

Primul ecran este ecranul care joacă rol de meniu, din care putem selecta butonul play pentru a ne conecta la camera de joc, ca în Figura 7.

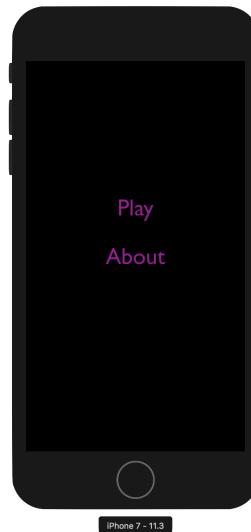


Figura 7 - Ilustrare meniu principal

Al doilea ecran este ecranul în care se desfășoară acțiunea jocului. În primă fază dacă jucătorul curent este primul venit în camera de joc, acesta are oportunitatea să pornească jocul prin apăsarea butonului Start, ca în Figura 8.

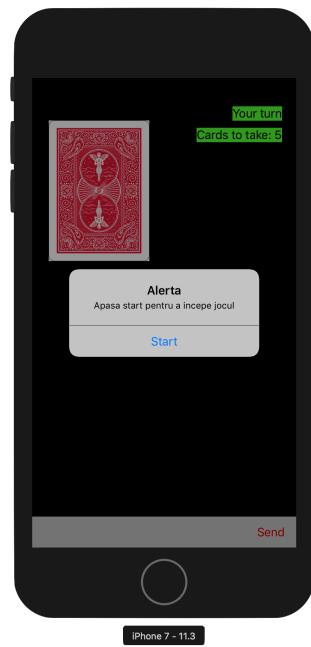


Figura 8 - Ilustrare incepere joc

După începerea jocului efectiv, jucătorul are posibilitatea să:

- Selecteze cărțile dorite și să apese Send pentru a le pune pe masă, atunci când este rândul său, ca în Figura 9.



Figura 9 - Ilustrare selectie cărți

- Selecteze alte cărți, în cazul în care a încercat să pună pe masă niște cărți care nu se pot da în această rundă, vezi Figura 10.



Figura 10 - Ilustrare selecție invalidă

- Apese pe cartea care este întoarsă cu spatele, care de fapt reprezintă pachetul de cărți și să ia o nouă carte, atunci când este rândul său.
- Aștepte, în cazul în care în acest moment nu este rândul său. Acest lucru se poate observa în Figura 11, unde eticheta Your turn este marcată cu roșu.

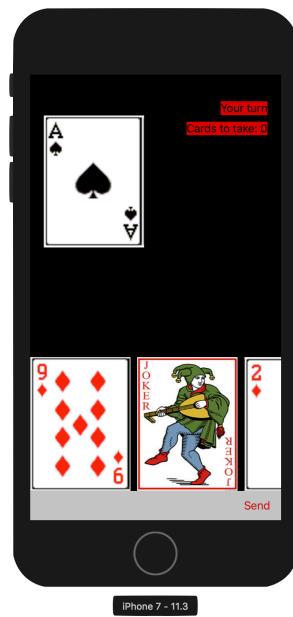


Figura 11 - Ilustrare așteptare rând la joc

4 Implementare

În acest capitol am să prezint abordarea tehnică și principalele funcționalități implementate, grupate pe fiecare componentă în parte.

4.1 Abordare tehnică

În acest subcapitol am să dau câteva informații referitoare la abordarea tehnică a acestui proiect din perspectiva configuratorului, serverului și al aplicației client.

4.1.1 Configurator mod de joc

Pentru modulul care se asigură de configurarea modului de joc, am ales să folosesc PHP deoarece oferă posibilitatea de a crea un formular și de a procesa datele astfel încât să se poată construi modul de joc ca și fișier indiferent de platformă prin intermediul APACHE.

4.1.2 Server

Am ales să construiesc serverul folosind Python 2.7, datorită faptului că mă adresez unei audiențe cu diferite preferințe legate de sistemul de operare. Astfel serverul poate fi rulat liniștit pe platforme precum Windows, Linux sau OS X.

Pe parcursul dezvoltării serverului s-au folosit module precum Socket pentru a asigura conexiunea client - server, Threading pentru a lucra cu fire de execuție și a pune la dispoziție un mecanism cu ajutorul căruia se pot face camere de joc, Random pentru funcția de amestecare care este necesară la gestionarea pachetelor de cărți și JSON pentru encodarea și decodarea mesajelor transmise.

4.1.3 Client

Am ales să construiesc aplicația client în Swift folosind XCode. Am folosit Swift în defavoarea Objective-C deoarece primul menționat este un limbaj modern care îmi oferă posibilitatea de a lucra mai eficient și de a trata erorile mai bine.

În cadrul aplicației client am folosit CocoaPods, un manager de dependențe, cu ajutorul căruia s-a făcut posibilă folosirea librăriei SocketSwift. Aceasta din urmă ne ajută să manipulăm socket-în ecosistemul iOS la un nivel înalt.

4.2 Implementare Configurator

Configuratorul este reprezentat de un program scris în PHP, care are la bază un formular web cu ajutorul căruia selectăm opțiunile dorite pentru a le transpune în modul de joc.

În primă fază, după completarea datelor în formular, se verifică cu ajutorul funcției din Figura 12 dacă datele sunt valide în vederea trecerii la pasul următor.

```

function check() {
    if ($_POST['GAME_MODE_NAME']) {
        echo "GameMode is OK.\n";
    } else {
        echo "[ERROR]Game Mode Name is not setted.\n";
        return False;
    }

    if ($_POST['NUMBER_OF_CARDS_PER_PLAYER']) {
        echo "Number of Cards per Player is OK.\n";
    } else {
        echo "[ERROR]Number of Cards per Player is not setted.\n";
        return False;
    }

    if ($_POST['MAX_NUMBER_OF_PLAYERS']) {
        echo "Maximum Number of Players is OK.\n";
    } else {
        echo "[ERROR]Maximum Number of Players is not setted.\n";
        return False;
    }

    if ($_POST['MIN_NUMBER_OF_PLAYERS']) {
        echo "Minimum Number of Players is OK.\n";
    } else {
        echo "[ERROR]Minimum Number of Players is not setted.\n";
        return False;
    }

    if ($_POST['MAX_NUMBER_OF_CARDS_IN_ONE_TURN']) {
        echo "Maximum Number of Cards in One Turn is OK.\n";
    } else {
        echo "[ERROR]Maximum Number of Cards in One Turn is not setted.\n";
        return False;
    }

    if ($_POST['PLAYERS_ORDER_RANDOM']) {
        echo "Players Order Random is OK.\n";
    } else {
        echo "[ERROR]Players Order Random is not setted.\n";
        return False;
    }

    return True;
}

```

Figura 12 - Ilustrare modalitate de a valida param. configurator

La al doilea pas, ilustrat în Figura 13, se adaugă în modul de joc anumite proprietăți simple precum GAME_MODE_NAME, NUMBER_OF_CARDS_PER_PLAYER și altele care nu necesită procesare.

```

11     $buffer = $buffer . "GAME_MODE_NAME" . $newline . $_POST['GAME_MODE_NAME'] . $newline;
12     $buffer = $buffer . "NUMBER_OF_CARDS_PER_PLAYER" . $newline . $_POST['NUMBER_OF_CARDS_PER_PLAYER'] . $newline;
13     $buffer = $buffer . "MAX_NUMBER_OF_PLAYERS" . $newline . $_POST['MAX_NUMBER_OF_PLAYERS'] . $newline;
14     $buffer = $buffer . "MIN_NUMBER_OF_PLAYERS" . $newline . $_POST['MIN_NUMBER_OF_PLAYERS'] . $newline;
15     $buffer = $buffer . "NUMBER_OF_CARDS_FOR_WIN" . $newline . $_POST['NUMBER_OF_CARDS_FOR_WIN'] . $newline;
16     $buffer = $buffer . "MAX_NUMBER_OF_CARDS_IN_ONE_TURN" . $newline . $_POST['MAX_NUMBER_OF_CARDS_IN_ONE_TURN'] . $newline;
17     $buffer = $buffer . "PLAYERS_ORDER_RANDOM" . $newline . strtoupper($_POST['PLAYERS_ORDER_RANDOM']) . $newline;
18

```

Figura 13 - Ilustrare initializare proprietăți simple în modul de joc

În continuare după cum se poate vedea în Figura 14, se adaugă proprietatea NOT_INITIAL_CARDS în modul de joc, proprietatea a căror date trebuie procesate în prealabil pentru a putea fi adăugate.

```

18      // not initial cards
19      if (isset($_POST['NOT_INITIAL_CARDS'])) {
20
21          $buffer = $buffer . "NOT_INITIAL_CARDS" . $newline;
22
23          $buffer = $buffer . "BEGIN" . $newline;
24
25          $not_initial_cards = $_POST['NOT_INITIAL_CARDS'];
26
27          foreach ($not_initial_cards as $card) {
28              if(array_search($card, $not_initial_cards) == sizeof($not_initial_cards)-1) {
29                  $buffer = $buffer . $card . $newline;
30              } else {
31                  $buffer = $buffer . $card . ", ";
32              }
33          }
34
35          $buffer = $buffer . "END" . $newline;
36      }
37
38
39

```

Figura 14 - Ilustrare procesare NOT_INITIAL_CARDS în configurator

În mod similar se întâmplă și la proprietatea care definește grupurile de cărți acceptate într-o singură tură.

În ceea ce privește proprietatea care definește acțiunile cărților, lucrurile se complică într-o anumită măsură, deoarece aceasta conține fiecare carte în parte, iar acestea din urmă conțin la rândul lor proprietăți cum ar fi cele de acceptare a anumitor cărți sau simboluri ulterioare, adăugare de cărți următorului jucător, blocarea unui număr de runde următorului jucător, stoparea anumitor acțiuni ale cărților date de jucătorul anterior și altele care se pot defini pe parcurs în funcție de preferințe. Acest concept, precum și alte detalii de implementare se pot vedea în Figura 15.

```

50 // cards actions
51
52 $buffer = $buffer . "CARDS_ACTIONS" . $newline;
53
54 for ($index = 1; $index <= 14; $index++) {
55
56     $buffer = $buffer . $index . $newline;
57
58     $buffer = $buffer . "BEGIN" . $newline;
59
60     $KEY_NEXT_CARDS_ACCEPTED = $index . "_NEXT_CARDS_ACCEPTED";
61     $KEY_ACCEPT_JUST_SAME_SYMBOL = $index . "_ACCEPT_JUST_SAME_SYMBOL";
62     $KEY_ADD_NEXT_PLAYER_CARDS = $index . "_ADD_NEXT_PLAYER_CARDS";
63     $KEY_REMOVE_NEXT_PLAYER_TURNS = $index . "_REMOVE_NEXT_PLAYER_TURNS";
64     $KEY_REMOVE_ADD_NEXT_PLAYER_CARDS = $index . "_REMOVE_ADD_NEXT_PLAYER_CARDS";
65
66     if(isset($_POST[$KEY_NEXT_CARDS_ACCEPTED]) && $_POST[$KEY_NEXT_CARDS_ACCEPTED]) {
67         $NEXT_CARDS_ACCEPTED = $_POST[$KEY_NEXT_CARDS_ACCEPTED];
68
69         $buffer = $buffer . "NEXT_CARDS_ACCEPTED" . $newline;
70         $buffer = $buffer . $NEXT_CARDS_ACCEPTED . $newline;
71     }
72
73     if(isset($_POST[$KEY_ACCEPT_JUST_SAME_SYMBOL]) && $_POST[$KEY_ACCEPT_JUST_SAME_SYMBOL]) {
74         $ACCEPT_JUST_SAME_SYMBOL = $_POST[$KEY_ACCEPT_JUST_SAME_SYMBOL];
75
76         $buffer = $buffer . "ACCEPT_JUST_SAME_SYMBOL" . $newline;
77         $buffer = $buffer . $ACCEPT_JUST_SAME_SYMBOL . $newline;
78     }
79
80     if(isset($_POST[$KEY_ADD_NEXT_PLAYER_CARDS]) && $_POST[$KEY_ADD_NEXT_PLAYER_CARDS]) {
81         $ADD_NEXT_PLAYER_CARDS = $_POST[$KEY_ADD_NEXT_PLAYER_CARDS];
82
83         $buffer = $buffer . "ADD_NEXT_PLAYER_CARDS" . $newline;
84         $buffer = $buffer . $ADD_NEXT_PLAYER_CARDS . $newline;
85     }
86
87     if(isset($_POST[$KEY_REMOVE_NEXT_PLAYER_TURNS]) && $_POST[$KEY_REMOVE_NEXT_PLAYER_TURNS]) {
88         $REMOVE_NEXT_PLAYER_TURNS = $_POST[$KEY_REMOVE_NEXT_PLAYER_TURNS];
89
90         $buffer = $buffer . "REMOVE_NEXT_PLAYER_TURNS" . $newline;
91         $buffer = $buffer . $REMOVE_NEXT_PLAYER_TURNS . $newline;
92     }
93
94     if(isset($_POST[$KEY_REMOVE_ADD_NEXT_PLAYER_CARDS]) && $_POST[$KEY_REMOVE_ADD_NEXT_PLAYER_CARDS]) {
95         $REMOVE_ADD_NEXT_PLAYER_CARDS = $_POST[$KEY_REMOVE_ADD_NEXT_PLAYER_CARDS];
96
97         $buffer = $buffer . "REMOVE_ADD_NEXT_PLAYER_CARDS" . $newline;
98         $buffer = $buffer . $REMOVE_ADD_NEXT_PLAYER_CARDS . $newline;
99     }
100
101    $buffer = $buffer . "END" . $newline;
102
103 }

```

Figura 15 - Ilustrare procesare CARDS_ACTIONS în configurator

4.3 Implementare Server

După cum a fost precizat și anterior, aplicația server este realizată folosind Python 2.7 și utilizează module standard precum *socket*, *sys*, *time*, *threading* și *json*.

Modulul *socket* oferă un mecanism de comunicare al mașinilor în rețea, folosind protocoale din stiva de protocoale TCP/IP.

În Figura 16 putem observa cum socket-ul este creat, legat cu hostul și portul, iar mai apoi acesta așteaptă conexiuni.

```

20     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21     print ('Socket created')
22
23     try:
24         s.bind((HOST, PORT))
25     except socket.error as msg:
26         print ('Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1])
27         sys.exit()
28
29     print ('Socket bind complete')
30
31     #Start listening on socket
32     s.listen(10)
33     print ('Socket now listening')

```

Figura 16 - Ilustrare folosire socket pe server

În Figura 17 putem observa cum serverul acceptă conexiunile și creează un nou fir de execuție căruia îi dă socketul clientului ca argument și o funcție ca și target.

```

270 ▼ while True:
271     print ("Server in asteptare ...")
272     connection, address = s.accept()
273     threading.Thread(target=gameHandler, args=(connection,)).start()
274     print ("While True end")
275

```

Figura 17 - Ilustrare așteptare clienți pe server

4.3.1 Implementare interpreter

În cadrul serverul a fost necesar implementarea unui interpreter care să preia modul de joc și să îl transforme în niște proprietăți specifice acestui joc de cărți generalizat. În cele ce urmează am să explic o parte din inițializarea interpreterului.

În Figura 18 putem observa cum se citește fișierul de configurare și se interpretează linie cu linie. Astfel se setează numele modului de joc, numărul de cărți de joc, dacă ordinea jucătorilor să fie aleatorie sau nu, numărul cărților pentru fiecare jucător și alte proprietăți care sunt similare și nu necesită o atenție deosebită.

```

15     def __init__(self, configFile):
16         config = open(configFile)
17         line = config.readline()
18
19         while line:
20
21             if line == "GAME_MODE_NAME\n":
22                 line = config.readline().replace("\n", "")
23                 self.GAME_MODE_NAME = line
24
25             elif line == "NUMBER_OF_CARDS\n":
26                 line = config.readline()
27                 self.NUMBER_OF_CARDS = int(line)
28
29             elif line == "PLAYERS_ORDER_RANDOM\n":
30                 line = config.readline()
31                 self.PLAYERS_ORDER_RANDOM = line.replace("\n", "")
32
33             elif line == "NUMBER_OF_CARDS_PER_PLAYER\n":
34                 line = config.readline()
35                 self.NUMBER_OF_CARDS_PER_PLAYER = int(line)
36
37
38

```

Figura 18 - Ilustrare inițializare interpreter

În Figura 19 se configurează parametrul care specifică ce grupuri de cărți se pot da într-o singură mână. După cum se observă în această situație există doi delimitatori și anume BEGIN și END pentru a fi clar când încep și când se termină instrucțiunile referitoare la această etichetă.

```

55 ▼           elif line == "ACCEPTED_CARDS_GROUPS_IN_ONE_TURN\n":
56               begin = config.readline()
57
58 ▼           if begin == "BEGIN\n":
59
60               line = config.readline()
61
62 ▼           while line != "END\n":
63               line = line.replace("\n", "")
64               self.ACCEPTED_CARDS_GROUPS_IN_ONE_TURN.append(line.split(","))
65               line = config.readline()

```

Figura 19 - Ilustrare managementul grupurilor de cărți în interpreter

În aceeași manieră, însă într-un mod mult mai complex, se configurează și acțiunile pe care le pot avea cărțile de joc. În Figura 20 observăm că între sintagmele BEGIN și END în acest moment putem găsi etichete referitoare la opțiuni de a adăuga cărți următorului jucător, de a bloca acțiunea următorului jucător sau chiar de a specifica ce fel de cărți se pot da după o anumită carte și dacă acestea trebuie să fie de același simbol sau nu.

```

67 ▼      elif line == "CARDS_ACTIONS\n":
68          card_number = config.readline()
69
70 ▼      while card_number:
71
72          card_actions_dictionary = {}
73
74          begin = config.readline()
75
76 ▼          if begin == "BEGIN\n":
77
78              line = config.readline()
79
80 ▼          while line != "END\n":
81
82              if line == "ADD_NEXT_PLAYER_CARDS\n":
83                  card_actions_dictionary["ADD_NEXT_PLAYER_CARDS"] = int(config.readline())
84
85              elif line == "REMOVE_NEXT_PLAYER_TURNS\n":
86                  card_actions_dictionary["REMOVE_NEXT_PLAYER_TURNS"] = int(config.readline())
87
88 ▼              elif line == "NEXT_CARDS_ACCEPTED\n":
89                  NEXT_CARDS_ACCEPTED_STRING = config.readline()
90                  NEXT_CARDS_ACCEPTED_STRING = NEXT_CARDS_ACCEPTED_STRING.replace("\n", "")
91                  NEXT_CARDS_ACCEPTED_LIST = NEXT_CARDS_ACCEPTED_STRING.split(",")
92                  card_actions_dictionary["NEXT_CARDS_ACCEPTED"] = NEXT_CARDS_ACCEPTED_LIST
93
94 ▼              elif line == "ACCEPT_JUST_SAME_SYMBOL\n":
95                  line = config.readline().replace("\n", "")
96                  ACCEPT_JUST_SAME_SYMBOL = line
97                  card_actions_dictionary["ACCEPT_JUST_SAME_SYMBOL"] = ACCEPT_JUST_SAME_SYMBOL
98
99                  line = config.readline()
100
101             self.CARDS_ACTIONS_DICTIONARY[card_number] = card_actions_dictionary
102             card_number = config.readline()
103

```

Figura 20 - Ilustrare management acțiuni cărți în interpreter

De asemenea, asemănător cum este inițializată proprietatea legată grupul de cărți de joc acceptate într-o singură mână în Figura 19, avem de această dată în Figura 21 felul cum se inițializează proprietatea legată de ce cărți nu pot fi puse jos în mod automat atunci când începe jocul. Dat fiind faptul că unele cărți au anumite funcții speciale în joc, este posibil ca prin modul de joc să nu dorim ca aceste tipuri de cărți să fie irosite la început de meci de către server.

```

104 ▼      elif line == "NOT_INITIAL_CARDS\n":
105          begin = config.readline()
106
107 ▼          if begin == "BEGIN\n":
108
109              line = config.readline()
110
111 ▼          while line != "END\n":
112              cardsArray = line.replace("\n", "").split(", ")
113
114              for card in cardsArray:
115                  self.NOT_INITIAL_CARDS.append(card)
116
117              line = config.readline()
118
119          line = config.readline()

```

Figura 21 - Ilustrare referitoare la cărțile ce nu pot fi inițiale în interpreter

Acestea fiind spuse, interpreterul joacă un rol foarte important în acest sistem deoarece face posibilă translatarea modului de joc dintr-un limbaj bazat pe token-valoare, un limbaj care poate fi scris foarte ușor sau chiar generat cu ajutorul configuratorului, în niște proprietăți specifice pentru un joc de cărți generalizat, proprietăți folosite ulterior în deciderea acțiunilor ce se pot face în acest joc.

4.3.2 Implementare mod de joc

O alta componentă importantă a serverului acestui joc de cărți generalizat este constituită de CustomGame, o clasă care se inițializează cu obiectul de tip Interpretor și o listă de socketi. Astfel această componentă transpune proprietățile stocate de către interpretor direct în reguli de joc pe care le folosește pentru a calcula stări valide, cărți valide pentru runda curentă, stări curente ale jucătorilor, funcționalități ale cărților și alte caracteristici definitorii.

În cele ce urmează am să explic anumite părți din clasa CustomGame, astfel încât să se înțeleagă modul de utilizare al acesteia în aplicația server.

În Figura 22 observăm ca este definită funcția cu ajutorul căreia se verifică dacă o stare, formată din cartea curentă de pe masă la un moment dat și o carte a unui jucător, constituie o stare validă care poate fi folosită pentru a avansa jocul. Astfel, după cum putem observă, se verifică dacă cartea curentă este în lista cărților acceptate de către cartea de pe masă sau dacă cartea de pe masă acceptă orice tip de carte, precum și dacă cartea de pe masă acceptă numai același simbol ca aceasta. În urma acestor verificări se poate concluziona dacă această stare, și anume cartea curentă peste cartea de pe masă este o stare validă. În caz afirmativ funcția va returna True, altfel False.

```

131 ▼   def isValidThisState(self, currentCard, playerCard):
132
133     currentCardNumber = ""
134     if len(currentCard) == 2:
135         currentCardNumber = currentCard[:1]
136     else:
137         currentCardNumber = currentCard[:2]
138
139     playerCardNumber = ""
140     if len(playerCard) == 2:
141         playerCardNumber = playerCard[:1]
142     else:
143         playerCardNumber = playerCard[:2]
144
145     # Cartile au acelasi numar, deci validam
146     if currentCardNumber == playerCardNumber:
147         return True
148
149     playerCardSymbol = playerCard[-1:]
150     currentCardSymbol = currentCard[-1:]
151
152 ▼   if currentCardNumber + "\n" in self.cardsActionsDictionary.keys():
153
154         cardDictionary = self.cardsActionsDictionary[currentCardNumber + "\n"]
155
156         if "NEXT_CARDS_ACCEPTED" in cardDictionary:
157             nextCardsAccepted = cardDictionary["NEXT_CARDS_ACCEPTED"]
158         else:
159             nextCardsAccepted = ["ALL"]
160
161         if "ACCEPT_JUST_SAME_SYMBOL" in cardDictionary:
162             acceptJustSameSymbol = cardDictionary["ACCEPT_JUST_SAME_SYMBOL"]
163         else:
164             acceptJustSameSymbol = "FALSE"
165
166     if nextCardsAccepted == ["ALL"] and acceptJustSameSymbol == "TRUE":
167         if playerCardSymbol == currentCardSymbol:
168             return True
169         else:
170             return False
171     elif nextCardsAccepted == ["ALL"] and acceptJustSameSymbol == "FALSE":
172         return True
173     elif acceptJustSameSymbol == "TRUE":
174         if playerCardNumber in nextCardsAccepted:
175             if playerCardSymbol == initialCardSymbol:
176                 return True
177     elif acceptJustSameSymbol == "FALSE":
178         if playerCardNumber in nextCardsAccepted:
179             return True
180
181     return False

```

Figura 22 - Ilustrare modalitate verificare carte în tură curentă

Cu ajutorul funcției descrise în Figura 23, utilizând și funcția definită anterior în Figura 22 putem să generăm o listă de cărți de joc valide pentru un anumit jucător, la un anumit moment astfel încât aplicația client să știe ce fel de cărți sunt permise să fie lăsate jos de către jucător la un anumit moment.

```

183 ▼    def getValidCardsForPlayer(self, player, currentCard):
184
185        validCards = list()
186
187 ▼    for playerCard in player.getCards():
188            if self.isValidThisState(currentCard, playerCard) == True:
189                validCards.append(playerCard)
190
191    return validCards
192

```

Figura 23 - Ilustrare modalitate verificare cărți valide în tură curentă

De asemenea, după cum putem observa în Figura 24, la rândul ei, funcția din Figura 23 este folosită pentru a putea genera o serie de grupuri de cărți valide la un moment dat pentru un anume jucător.

```

192
193    def getValidGroupsOfCardsForPlayer(self, player, currentCard):
194
195        validGroupsOfCards = list()
196
197        for group in self.acceptedCardsGroupsInOneTurn:
198
199            newGroup = list()
200
201            for card in player.getCards():
202                if card in group:
203                    newGroup.append(card)
204
205            if len(newGroup) > 1:
206                mach = False
207                for card in newGroup:
208                    if self.isValidThisState(currentCard, card) == True:
209                        validGroupsOfCards.append(newGroup)
210                        break
211
212    return validGroupsOfCards
213

```

Figura 24 - Ilustrare modalitate identificare grupuri valide de cărți în runda curentă

În aceeași manieră sunt construite și alte funcții și componente ale serverului, plecând de la o anumită funcționalitate de bază să o putem extinde astfel încât să nu fim puși în postură de a duplica codul.

Ca urmare a funcțiilor prezentate în acest capitol, putem concretiza rezultatele acestora în Figura 25, unde este ilustrat modul în care acestea se folosesc pentru a putea genera starea curentă la un moment dat pentru un jucător. Această parte este foarte importantă deoarece reprezintă fundamentalul acțiunilor ulterioare pe care jucătorul le poate face în joc. Această stare conține informații legate de cartea curentă de pe masă, dacă este sau nu rândul jucătorului în cauză, cărțile disponibile, dacă jucătorul poate sau nu să ia o carte de jos, dacă jucătorul este blocat sau nu de către alt jucător în următoarele runde, ce cărți sau grupuri de cărți sunt valide pentru această rundă, dacă jucătorul are de luat cărți și alte posibile informații care se pot adăuga în funcție de necesitățile ulterioare datorate dezvoltării sistemului.

```

214 def getCurrentStateForPlayer(self, player):
215
216     if player.isYourTurn() == True:
217         if player.getWaitTimes() == 0 and len(self.cardsArray) > 0:
218             player.setCanGetNewCard(True)
219         else:
220             player.setCanGetNewCard(False)
221     else:
222         player.setCanGetNewCard(False)
223
224     state = {}
225     state['currentCard'] = self.currentCard
226     state['isYourTurn'] = player.isYourTurn()
227     state['cards'] = player.getCards()
228
229     state['canGetNewCard'] = player.getCanGetNewCard()
230
231     state['waitTimes'] = player.getWaitTimes()
232     state['validCards'] = self.getValidCardsForPlayer(player, self.currentCard)
233     state['validGroupsOfCards'] = self.getValidGroupsOfCardsForPlayer(player, self.currentCard)
234     state['cardsToTake'] = player.getCardsToTake()
235     state['code'] = "2" # in game code
236
237     return state
238

```

Figura 25 - Ilustrare calcularea stării curente pentru un anumit jucător

4.3.3 Implementare sistem camere

Mecanismul de management al camerelor de joc are responsabilitatea de distribui jucătorii în camere de joc, iar atunci când acestea ating numărul maximum de jucători, acest mecanism trebuie să creeze noi camere pentru a le pune la dispoziția noilor jucători.

Raționamentul descris mai sus îl avem de această dată exemplificat în Figura 26, unde observăm ca în momentul când nu este nicio cameră se creează una nouă, iar jucătorul preia statutul de gazdă și poate ulterior să înceapă jocul. În cazul în care camerele disponibile sunt pline, atunci serverul creează o nouă cameră pentru jucătorul curent, altfel jucătorul este adăugat la ultima camera creată, după cum putem vedea la linia 126.

```

124 if len(rooms) > 0:
125     if len(rooms[-1]) < interpreter.getMaxNumberOfPlayers():
126         rooms[-1].append(connection)
127     else:
128         newRoom = list()
129         newRoom.append(connection)
130         rooms.append(newRoom)
131         indexOfRoom = rooms.index(newRoom)
132
133         threading.Thread(target=startGame, args=(indexOfRoom,)).start()
134     else:
135         newRoom = list()
136         newRoom.append(connection)
137         rooms.append(newRoom)
138         indexOfRoom = rooms.index(newRoom)
139
140         threading.Thread(target=startGame, args=(indexOfRoom,)).start()
141

```

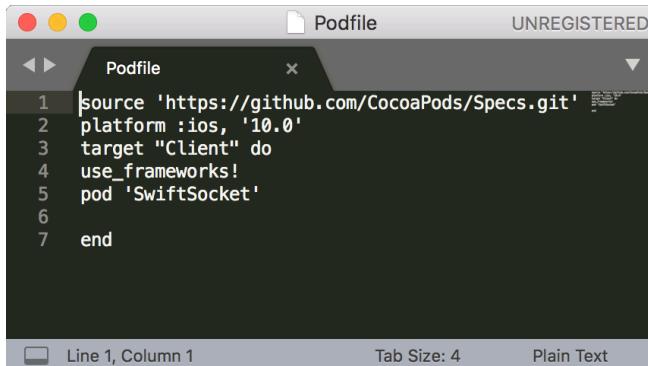
Figura 26 - Ilustrare management camere de joc

4.4 Implementare Client

După cum am văzut anterior, pe parte de client avem o aplicație mobilă pentru sistemul de operare iOS, dezvoltată folosind XCode în limbajul de programare Swift. În acest subcapitol am să parcurg părțile care mi s-au părut relevante din implementarea aplicației client.

4.4.1 Configurare și actualizare dependențe externe

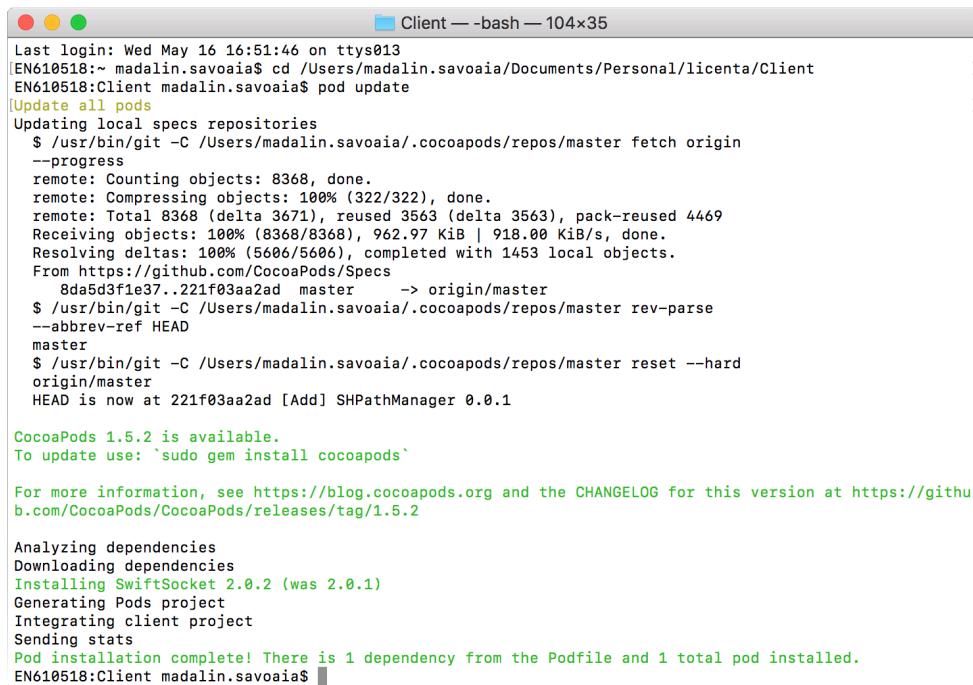
Managementul dependențelor externe este asigurat de cocoapods, cu ajutorul căruia am setat librăria SwiftSocket ca și dependență în fișierul Podfile, ca în exemplul din Figura 27.



```
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '10.0'
target "Client" do
use_frameworks!
pod 'SwiftSocket'
end
```

Figura 27 - Ilustrare configurare dependențe cocoapods

Astfel ne este oferită posibilitatea de a actualiza dependențele într-un mod eficient și simplu printr-o comandă în terminal, ca în exemplul prezentat în Figura 28.



```
Last login: Wed May 16 16:51:46 on ttys013
[EN610518:~ madalin.savoaia$ cd /Users/madalin.savoaia/Documents/Personal/licenta/Client
[EN610518:Client madalin.savoaia$ pod update
[Update all pods
Updating local specs repositories
$ /usr/bin/git -C /Users/madalin.savoaia/.cocoapods/repos/master fetch origin
--progress
remote: Counting objects: 8368, done.
remote: Compressing objects: 100% (322/322), done.
remote: Total 8368 (delta 3671), reused 3563 (delta 3563), pack-reused 4469
Receiving objects: 100% (8368/8368), 962.97 KiB | 918.00 KiB/s, done.
Resolving deltas: 100% (5606/5606), completed with 1453 local objects.
From https://github.com/CocoaPods/Specs
  8da5d3f1e37..221f03aa2ad  master      -> origin/master
$ /usr/bin/git -C /Users/madalin.savoaia/.cocoapods/repos/master rev-parse
--abbrev-ref HEAD
master
$ /usr/bin/git -C /Users/madalin.savoaia/.cocoapods/repos/master reset --hard
origin/master
HEAD is now at 221f03aa2ad [Add] SHPathManager 0.0.1

CocoaPods 1.5.2 is available.
To update use: `sudo gem install cocoapods`


For more information, see https://blog.cocoapods.org and the CHANGELOG for this version at https://github.com/CocoaPods/CocoaPods/releases/tag/1.5.2

Analyzing dependencies
Downloading dependencies
Installing SwiftSocket 2.0.2 (was 2.0.1)
Generating Pods project
Integrating client project
Sending stats
Pod installation complete! There is 1 dependency from the Podfile and 1 total pod installed.
[EN610518:Client madalin.savoaia$ ]
```

Figura 28 - Ilustrare actualizare dependențe cocoapods

4.4.2 Implementare interfață grafică

Aplicația client este constituită din două ecrane, după cum putem observa în Figura 29, primul este reprezentat de meniul principal din care se poate selecta opțiunea de a începe un joc, loc unde se pot adăuga de asemenea și alte legături către funcționalități ulterioare cum ar fi clasamente, istorice, și al doilea ecran care este reprezentat de jocul în sine.

În această secțiune este important de specificat faptul că pentru a dezvolta interfața grafică am folosit Storyboards⁸.

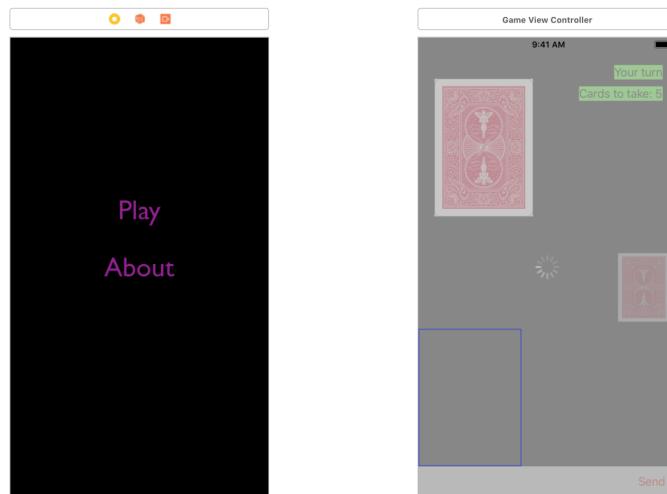


Figura 29 - Ilustrare ecran meniu și ecran joc în Storyboard

4.4.3 Implementare comunicație cu serverul

În timpul jocului, când un jucător selectează o anumită carte sau un grup de cărți și încearcă să le trimită la server, atunci intervin câteva verificări la nivel de client.

După cum se poate observa în Figura 30 se începe cu verificarea dacă este rândul jucătorului, dacă cartea sau grupul de cărți sunt valide pentru starea curentă, iar mai apoi se trimite rezultatul la server. În caz că nu se respectă cel puțin una dintre condiții, aplicația va afișa o eroare jucătorului.

⁸ Componentă din UIKit care facilitează lucrul cu interfețe grafice și legarea acestora de funcționalități din cod.

```

225     if self.isYourTurn == false {
226         return
227     }
228
229     if selectedCards.count == 1 {
230         guard let card = self.selectedCards.first else { return }
231         if isCardValid(card: card) == true {
232             self.selectedCards.removeAll()
233             self.sendMessageToString(string: card)
234             self.collectionView.reloadData()
235         } else {
236             showMessage(title: "Atentie", message: Constants.noValidCardMessage, dismiss: false)
237         }
238     } else if selectedCards.count > 0 {
239         var isSelectionSent = false
240         for group in validGroupsOfCardsForCurrentTurn {
241             var isValidSelection = true
242             var messageToString = ""
243             for card in selectedCards {
244                 if group.contains(card) {
245                     messageToString += card
246                 } else {
247                     isValidSelection = false
248                     break
249                 }
250                 if let index = selectedCards.index(of: card) {
251                     // Daca nu este ultima carte
252                     if index < selectedCards.count - 1 {
253                         messageToString += ","
254                     }
255                 }
256             }
257             if isValidSelection == true {
258                 sendMessageToString(string: messageToString)
259                 isSelectionSent = true
260                 break
261             }
262         }
263         if isSelectionSent {
264             // selectia a fost trimisa la server
265             self.selectedCards.removeAll()
266             print("Trimitem Grupul de carti")
267             self.collectionView.reloadData()
268         } else {
269             // selectia nu a fost trimisa la server
270             print("grupul de carti nu este valid")
271             showMessage(title: "Atentie", message: Constants.noValidGroupCardsMessage, dismiss: false)
272         }
273     } else {
274         showMessage(title: "Atentie", message: Constants.noCardSelectedMessage, dismiss: false)
275     }
276 }

```

Figura 30 - Ilustrare verificare trimitere carte sau grup de cărți de la client la server

5 Manual de utilizare

5.1 Manual de utilizare Configurator

În acest capitol am să exemplific cum se folosește configuratorul pentru a genera un mod de joc.

5.1.1 Generare mod de joc

La acest pas trebuie să accesăm configuratorul, ca în Figura 31 și să setăm în primul rând valorile atributelor standard următoare:

- Game Mode Name - reprezintă numele modului de joc
- Number of Cards per Player - reprezintă numărul cărților pentru fiecare jucător
- Maximum Number of Players - reprezintă numărul maximum de jucători
- Minimum Number of Players - reprezintă numărul minimum de jucători
- Number of Cards for Win - reprezintă pragul de terminare a jocului
- Maximum Number of Cards in One Turn - reprezintă numărul maximum de cărți permise într-o singură tură
- Player Order Random - configerează dacă jucătorii să fie în ordine aleatorie sau nu
- Select not initial cards - prin acest câmp setăm ce cărți nu pot fi folosite drept carte inițială la începerea jocului
- Accepted Cards Groups - în aceste câmpuri setăm ce fel de grupuri se pot forma cu cărțile, astfel încât să le putem da într-o singură tură. Spre exemplu un grup poate fi constituit din toate cărțile cu valoarea 10, acest lucru fiind exprimat prin 101,102,103,104 . În mod similar se pot crea și alte grupuri pentru alte cărți.

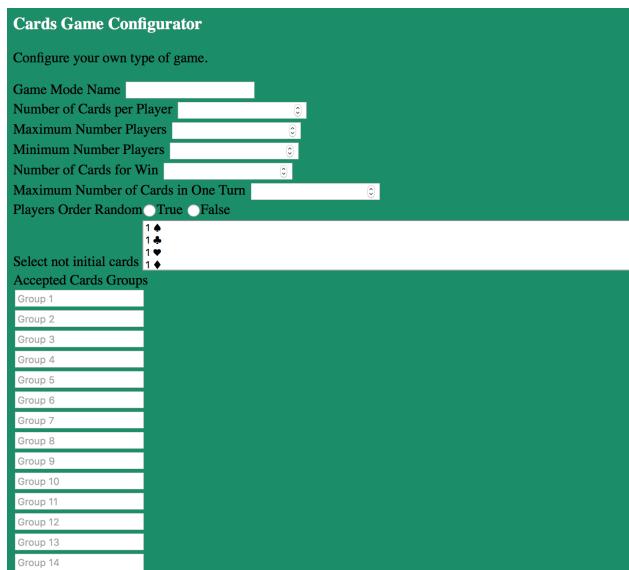


Figura 31 - Ilustrare configurare mod de joc, partea 1

La pasul al doilea trebuie să setăm acțiunile ce le pot avea cărțile de joc. Spre exemplu, lunând în calcul cartea cu numărul 1 în toate variantele ei, putem configura următoarele (vezi Figura 32) :

- Next Cards Accepted - permite setarea cărților acceptate peste cartea curentă. Dacă se dorește acceptarea tuturor cărților se setează ALL, iar dacă spre exemplu se dorește acceptarea cărților cu numărul doi și simbolurile treflă și inimă se setează 22, 23 .
- Accept just same Symbol - este util atunci când am setat câmpul precedent ca fiind ALL. Astfel putem să acceptăm toate cărțile cu același simbol sau chiar și indiferent de acesta.
- Add Next Player Cards - oferă posibilitatea cărții curente de a adăuga un număr finit de cărți următorului jucător
- Remove Next Player Turns - oferă posibilitatea cărții curente de a bloca jucătorul următor pentru un număr finit de runde.
- Remove Add Next Player Cards - oferă posibilitatea cărții curente de a bloca acțiunea de Add Next Player Cards a cărții anterioare

<img alt="Screenshot of the 'Cards Actions' configuration interface. It shows three sections for card values 1, 2, and 3, each with four configuration options: 'Next Cards Accepted' (checkboxes for ALL, 22, 23), 'Accept just same Symbol' (radio buttons for YES or NO), 'Add Next Player Cards' (checkbox with a dropdown menu showing 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157,

```

GAME_MODE_NAME
Macaua Vers. 2
NUMBER_OF_CARDS_PER_PLAYER
5
MAX_NUMBER_OF_PLAYERS
5
MIN_NUMBER_OF_PLAYERS
2
NUMBER_OF_CARDS_FOR_WIN
0
MAX_NUMBER_OF_CARDS_IN_ONE_TURN
4
PLAYERS_ORDER_RANDOM
FALSE
NOT_INITIAL_CARDS
BEGIN
11, 12, 13, 14
21, 22, 23, 24, 31, 32, 33, 34, 41, 42, 43, 44, 71, 72, 73, 74, 141, 142, 143, 144
END
ACCEPTED_CARDS_GROUPS_IN_ONE_TURN
BEGIN
11,12,13,14
21,22,23,24,31,32,33,34,141,142,143,144
41,42,43,44
51,52,53,54
61,62,63,64
71,72,73,74
81,82,83,84
91,92,93,94
101,102,103,104
111,112,113,114
121,122,123,124
131,132,133,134
END

```

Figura 33 - Mod de joc, partea 1

```

CARDS_ACTIONS
1
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
2
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
ADD_NEXT_PLAYER_CARDS
2|
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
3
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
ADD_NEXT_PLAYER_CARDS
3
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
4

```

Figura 34 - Mod de joc, partea 2

5.2 Manual de utilizare Server

În acest capitol am să prezint câteva informații referitoare la cum se adaugă un mod de joc, cum se configurează serverul și cum se rulează acesta.

5.2.1 Exemplificare adăugare mod de joc

După ce modul de joc a fost creat în prealabil acesta trebuie adăugat pe server în directorul GameModes.

În continuare, pentru ca acesta să fie folosit, trebuie să accesăm fișierul server.py, iar la începutul acestuia se poate identifica variabila gameModeFileName, pe care trebuie să o inițializăm cu numele fișierului.

5.2.2 Exemplificare configurare server

În ceea ce privește partea de configurare a serverului, aceasta se realizează într-un mod simplu. Se deschide fișierul server.py din directorul serverului și se modifică variabila HOST cu adresa ip a mașinii pe care o să rulați serverul, iar variabila PORT cu portul la care doriți să fie disponibil serviciul.

5.2.3 Exemplificare rulare server

Pentru a rula serverul este necesar să deschideți directorul acestuia și să executați fișierul server.py. În caz că folosiți linux sau os x, executarea se poate face cu ajutorul comenzi *python server.py*.

5.3 Manual de utilizare Client

În acest capitol am să prezint câteva informații utile referitoare la procesul de configurare, rulare și interacționare în ceea ce privește aplicația client.

5.3.1 Exemplificare configurare client

Pentru a configura aplicația client, se deschide fișierul Client.xcworkspace cu ajutorul programului XCode.

În continuare se deschide directorul Modules, urmat de Game, iar apoi de Controllers în care putem găsi GameViewController.swift. În acesta se modifică variabila host cu adresa ip a mașinii care găzduiește serverul și variabila port cu portul la care se găsește serviciul.

5.3.2 Exemplificare rulare client

După pasul de configurare al aplicației client, presupunând că aplicația XCode este deschisă de la pasul precedent, se selectează din bara de instrumente de sus un dispozitiv fizic conectat la calculator sau un simulator și se apasă combinația de taste CMD + R.

5.3.3 Exemplificare începere joc

Pentru a începe o rundă de joc, trebuie apăsat butonul play, ca în Figura 35, iar apoi trebuie confirmat că doriți să înceapă jocul apăsând butonul Start, din Figura 36.



Figura 35 - Ilustrare meniu principal client

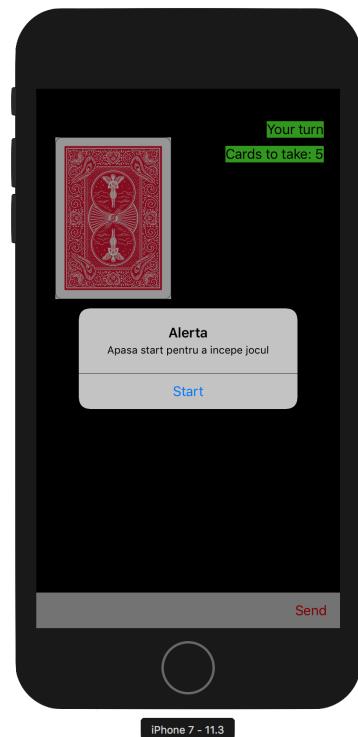


Figura 36 - Ilustrare începere joc client

5.3.4 Exemplificare comenzi joc

În timpul unei runde de joc începute, atunci când eticheta din colțul din dreapta al ecranului, care se numește *Your turn* este de culoare verde, avem certitudinea că putem să efectuăm o comandă.

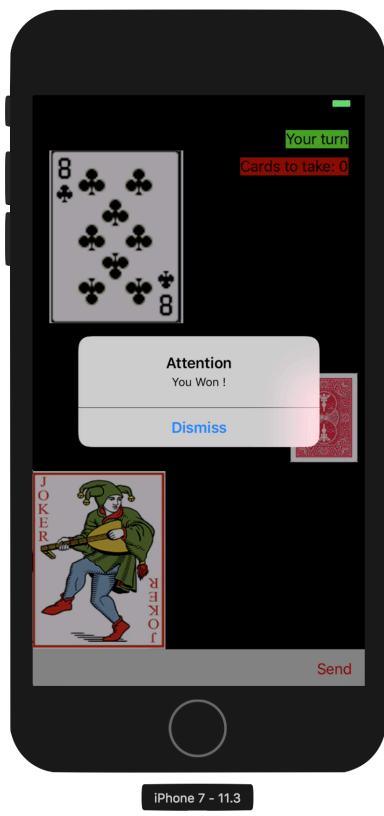
Una dintre comenziile disponibile este selecția unei cărți sau al unui grup de cărți și punerea pe masă peste cartea curentă. O ilustrație cu privire la această comandă avem spre exemplu în capitolul Interfață cu utilizatorul la Figura 9 - Ilustrare selectie cărți **Error! Reference source not found.**

O altă comandă disponibilă este luarea unei cărți din pachetul de cărți, care de asemenea se poate face apăsând pe cartea întoarsă din dreptul ecranului, aşa cum putem vedea în ilustrația comenzi precedente.

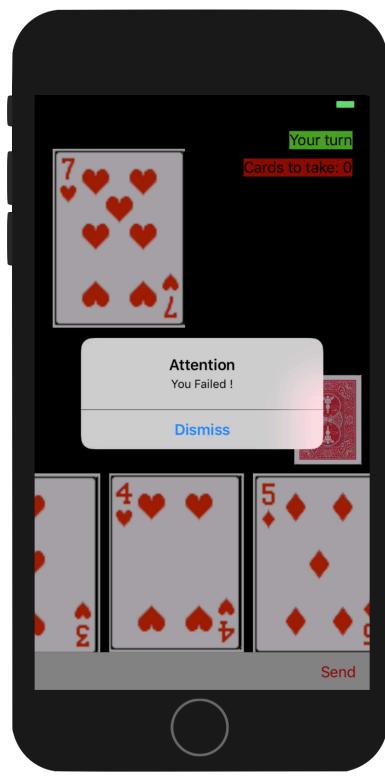
5.3.5 Exemplificare încheiere joc

Jocul se încheie pentru un jucător câștigător atunci când acesta atinge pragul de terminare setat în modul de joc, adică spre exemplu în modul de joc macao, atunci când acesta termină toate cărțile. O ilustrație din aplicație pentru acest exemplu este disponibilă în Figura 37.

În caz că jucătorul curent este înfrânt, de asemenea acesta va fi notificat în aplicație prin intermediul unui mesaj, un bun exemplu este în Figura 38.



iPhone 7 - 11.3



iPhone 7 - 11.3

Figura 37 - Ilustrare încheiere joc câștigător

Figura 38 - Ilustrare încheiere joc pierzător

6 Concluzii

Având în vedere aspectele prezentate pe parcursul acestei lucrări, putem concluziona faptul că am îndeplinit punctele stabilite anterior, iar eventualii dezvoltatori care aleg să folosească această soluție, au parte de un punct de plecare care să le faciliteze debutul în crearea unui joc de cărți generalizat.

Din punctul meu de vedere, cea mai complexă parte a acestui proiect până în acest moment a fost cea în care s-a construit și implementat limbajul prin care se pot face moduri de joc. Limbajul în sine nu este foarte complicat de construit, însă este un consumator considerabil de timp a construind un interpreter și a ține cont de configurația dată de limbaj.

Pe viitor această soluție poate fi îmbunătățită prin următoarele:

- modificarea interfeței grafice de pe parte de client astfel încât să fie mai prietenoasă pentru utilizatori
- posibilitatea de a stoca date despre jocurile de cărți precedente
- adăugare statistici și topuri disponibile jucătorilor
- adăugare posibilitate de a configura un mod de joc în aplicație în vederea folosirii într-o cameră specifică de către utilizator
- adăugarea unei funcționalități prin care un jucător să invite un altul la joc, cel din urmă primind în acest caz o notificare

7 Anexe

```
GAME_MODE_NAME
Macau Vers. 2
NUMBER_OF_CARDS_PER_PLAYER
MAX_NUMBER_OF_PLAYERS
5
MIN_NUMBER_OF_PLAYERS
2
NUMBER_OF_CARDS_WIN
MAX_NUMBER_OF_CARDS_IN_ONE_TURN
4
PLAYERS_ORDER_RANDOM
FALSE
NOT_INITIAL_CARD
BEGIN
    12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, 41, 42, 43, 44, 71, 72, 73, 74, 141, 142, 143, 144
END
ACCEPTED_CARDS_GROUPS_IN_ONE_TURN
11,-12,-13,-14
21,22,23,24,31,32,33,34,141,142,143,144
41,42,43,44
51,52,53,54
61,62,63,64
71,72,73,74
81,82,83,84
91,92,93,94
101,102,103,104
111,112,113,114
121,122,123,124
131,132,133,134
CARDS_ACTIONS
1
REIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
```

Figura 39 - Fișier mod de joc generat, partea 1

```
2 BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
ADD_NEXT_PLAYER_CARDS
2
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
3
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
ADD_NEXT_PLAYER_CARDS
3
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
4
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
REMOVE_NEXT_PLAYER_TURNS
1
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
5
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
TRUE
END
```

Figura 40 - Fișier mod de joc generat, partea 2

```
6
BEGIN
    ACCEPT_NEXT_CARDS_ACCEPTED
    ALL
    ACCEPT__JUST__SAME_SYMBOL
    REMOVE_ADD_NEXT_PLAYER_CARDS
    FALSE
END
B
BEGIN
    ACCEPT_NEXT_CARDS_ACCEPTED
    ALL
    ACCEPT__JUST__SAME_SYMBOL
    REMOVE_ADD_NEXT_PLAYER_CARDS
    FALSE
END
B
BEGIN
    ACCEPT_NEXT_CARDS_ACCEPTED
    ALL
    ACCEPT__JUST__SAME_SYMBOL
    REMOVE_ADD_NEXT_PLAYER_CARDS
    FALSE
END
B
BEGIN
    ACCEPT_NEXT_CARDS_ACCEPTED
    ALL
    ACCEPT__JUST__SAME_SYMBOL
    REMOVE_ADD_NEXT_PLAYER_CARDS
    TRUE
END
```

Figura 41 - Fișier mod de joc generat, partea 3

```
10 BEGIN  
NEXT_CARDS_ACCEPTED  
ALL  
ACCEPT_JUST_SAME_SYMBOL  
TRUE  
REMOVE_ADD_NEXT_PLAYER_CARDS  
FALSE  
END  
11 BEGIN  
NEXT_CARDS_ACCEPTED  
ALL  
ACCEPT_JUST_SAME_SYMBOL  
TRUE  
REMOVE_ADD_NEXT_PLAYER_CARDS  
FALSE  
END  
12 BEGIN  
NEXT_CARDS_ACCEPTED  
ALL  
ACCEPT_JUST_SAME_SYMBOL  
TRUE  
REMOVE_ADD_NEXT_PLAYER_CARDS  
FALSE  
END  
13 BEGIN  
NEXT_CARDS_ACCEPTED  
ALL  
ACCEPT_JUST_SAME_SYMBOL  
TRUE  
REMOVE_ADD_NEXT_PLAYER_CARDS  
FALSE  
END  
14 BEGIN  
NEXT_CARDS_ACCEPTED  
ALL  
ACCEPT_JUST_SAME_SYMBOL  
FALSE
```

Figura 42 - Fișier mod de joc generat, partea 4

```
Macaua_v2.txt

REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
11
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
12
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
13
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
TRUE
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
14
BEGIN
NEXT_CARDS_ACCEPTED
ALL
ACCEPT_JUST_SAME_SYMBOL
FALSE
ADD_NEXT_PLAYER_CARDS
19
REMOVE_ADD_NEXT_PLAYER_CARDS
FALSE
END
```

Figura 43 - Fișier mod de joc generat, partea 5

8 Bibliografie

1. David Ascher, Mark Lutz (2009). Learning Python
2. Brandon Rhodes, John Goerzen (2014). Foundations of Python Network Programming
3. Apple Inc. (2014). The Swift Programming Language