

Guess Card Game

СЪДЪРЖАНИЕ (С ЛИНКОВЕ)

1. Описание	1
2. Технически предпоставки	1
3. Структури в проекта	4
4. Стартиране на сървъра	6
5. Автентикация	6
6. Валидация	6
7. Нормален ход на играта	11
8. Коментари	17

1. Описание

Guess Card е проста хазартна игра с карти, която наподобява играта „Война“ или „Бакара“ но с по една карта - за играла и за дилъра и без второ теглене при конкретни условия. Целта на играта е да се предвиди изходът от това дали картата, която ще изтегли дилъра е с по-висока или по-ниска номинална стойност от картата която е изтеглил играла.

Картите имат следните стойности:

- Асата са най-силните карти и се равняват на 14 точки всяко.
(В случай, че искаме това да се преправи е необходимо само да преименуваме имената им например от 14_Ace до 01_Ace в пакета "com/cayetano/engine/Engine.java" както и конкретните имена на картите от снимките в resources папката)
- Картите от 2 до 10 се равняват на числото, изписано на лицето на картата.
- Валетата, дамите и поповете се равняват съответно на 11, 12 и 13 точки.
- Жокерите се използват за карти по подразбиране или разделител. Например, поставяме жокер за карта на играла когато играла все още не е изтеглил карта от тестето. Поставяме жокер за карта на дилъра, когато все още не е направено предположение каква ще е неговата карта или картите в тестето са свършили.

Само числовата стойност на всяка карта е от значение в основната игра. Всеки цвят на картите (купи, спатии, пики или кари) не е от значение.

Първоначално трябва да се изтегли карта от тестето с карти и да се направи залог в секцията на дилъра. След това трябва да се отгатне дали картата на дилъра ще е с по-висока или по-ниска стойност. При познаване - залогът се удвоява. Ако непознаете - залогът се губи. При равни числови стойности - залогът остава непроменен и се връща обратно в баланса на играла (в случая нито сте спечелили нито загубили, но в секцията "last win" се отбелязва стойността на залога). Повече описание може да намерите в точка 5.

2. Технически предпоставки

Сървърът на приложението е разработен с **Java 19**:

openjdk 19.0.2 2023-01-17
OpenJDK Runtime Environment (build 19.0.2)

Приложението се свързва с MySQL база от данни в която се пази състоянието на играта (минималната информация необходима за пълното възстановяване на играта, в случай, че клиента се logout-не или настъпи някакво външно събитие като спиране на тока например). В конфигурационният файл на Spring приложението (/src/main/resources/application.yml) са описани параметрите необходими за тази свръзка:

- името на драйвъра за свръзка с базата
- пътя до базата
- потребителско име с което се свързвате с базата
- парола
- диалект, с който hibernate ще изпълнява заявките към базата

Текущите настройки са следните:

MySQL

```
spring:  
  datasource:  
    driverClassName: com.mysql.cj.jdbc.Driver  
    url: jdbc:mysql://localhost:3306/guess_card?  
    allowPublicKeyRetrieval=true&createDatabaseIfNotExist=true&useUnicode=true&characterEncoding=UTF-8  
    username: root  
    password: root  
  jpa:  
    database-platform: org.hibernate.dialect.MySQLDialect
```

След въпросителният знак на url-а следват опционални настройки на базата. В случая параметърът `createDatabaseIfNotExist=true` ще работи само за MySQL база от данни. Ако например приложението се свързва с PostgreSQL или MSSQL Server база от данни, е необходимо да промените конфигурациите по следния начин (съответно):

PostgreSQL

```
spring:  
  datasource:  
    driverClassName: org.postgresql.Driver  
    url: jdbc:mysql://localhost:5432/guess_card  
    username: your_db_username_here  
    password: your_db_password_here  
  jpa:  
    database-platform: org.hibernate.dialect.PostgreSQLDialect
```

MSSQL Server

```
spring:  
  datasource:  
    driverClassName: com.mysql.cj.jdbc.Driver  
    url: jdbc:sqlserver://localhost;databaseName=guess_card  
    username: your_db_username_here  
    password: your_db_password_here  
  jpa:  
    database-platform: org.hibernate.dialect.SQLServer2012Dialect
```

И в двета случая на PostgreSQL и MSSQL Server, е необходимо преди това да е създадена ръчно базата от данни с име **guess_card**, докато текущият вариант с MySQL създава базата автоматично, ако не е създадена.

Параметрите по-долу от конфигурацията, казват на Hibernate, че искаме автоматично да създадем таблиците в базата спрямо Java Entity-тата в приложението.

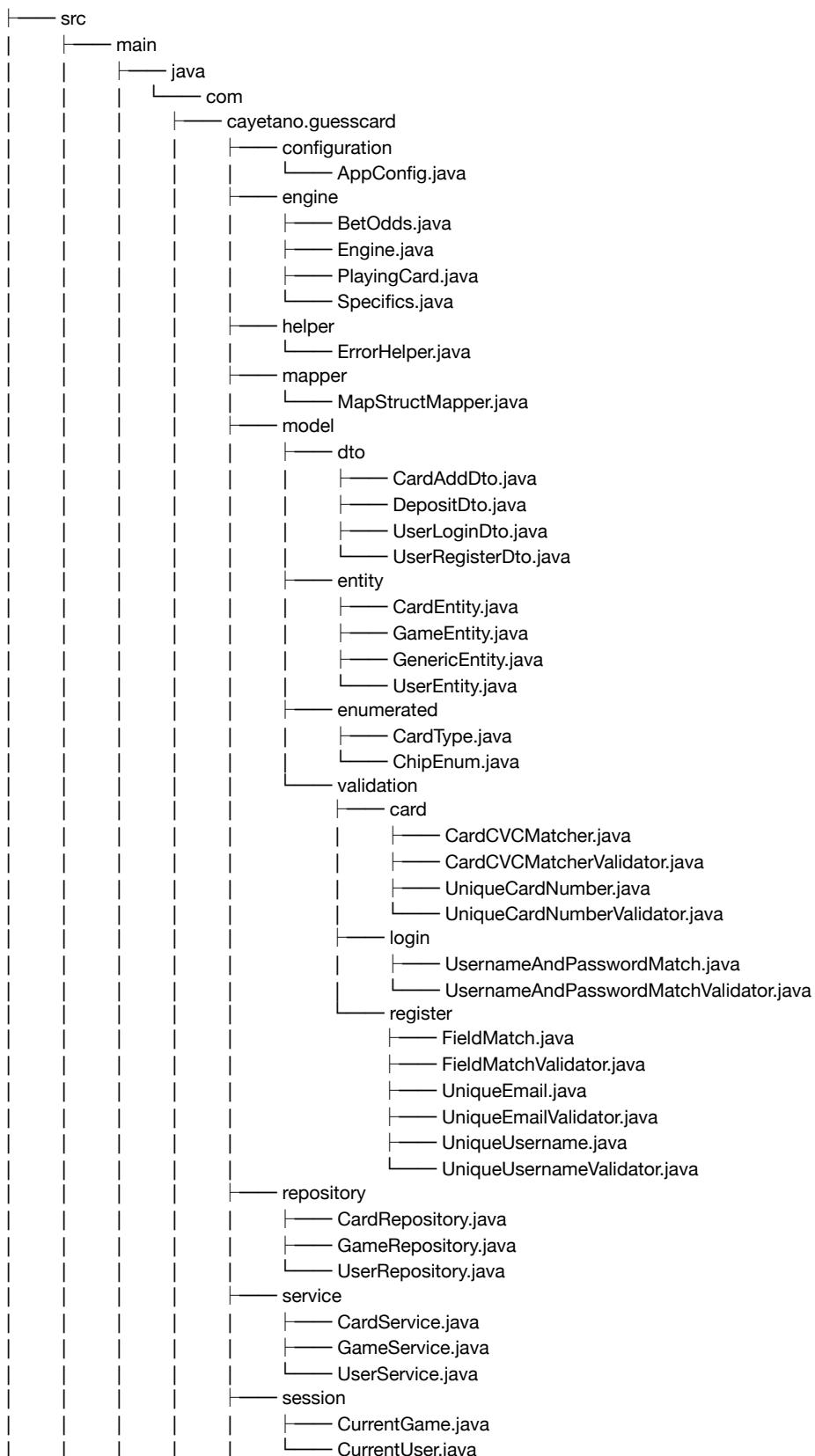
```
  hibernate:  
    ddl-auto: update
```

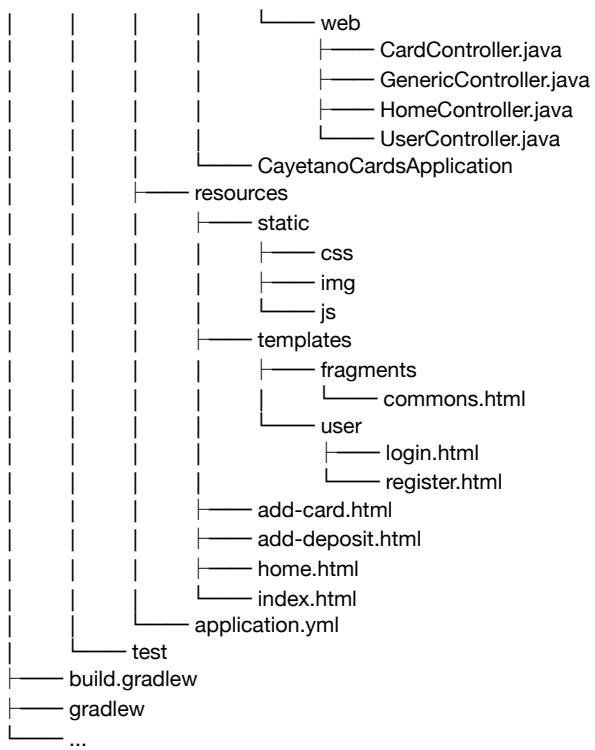
Проектът използва **gradle** инструмент за автоматизирано изграждане.

3. Структури в проекта

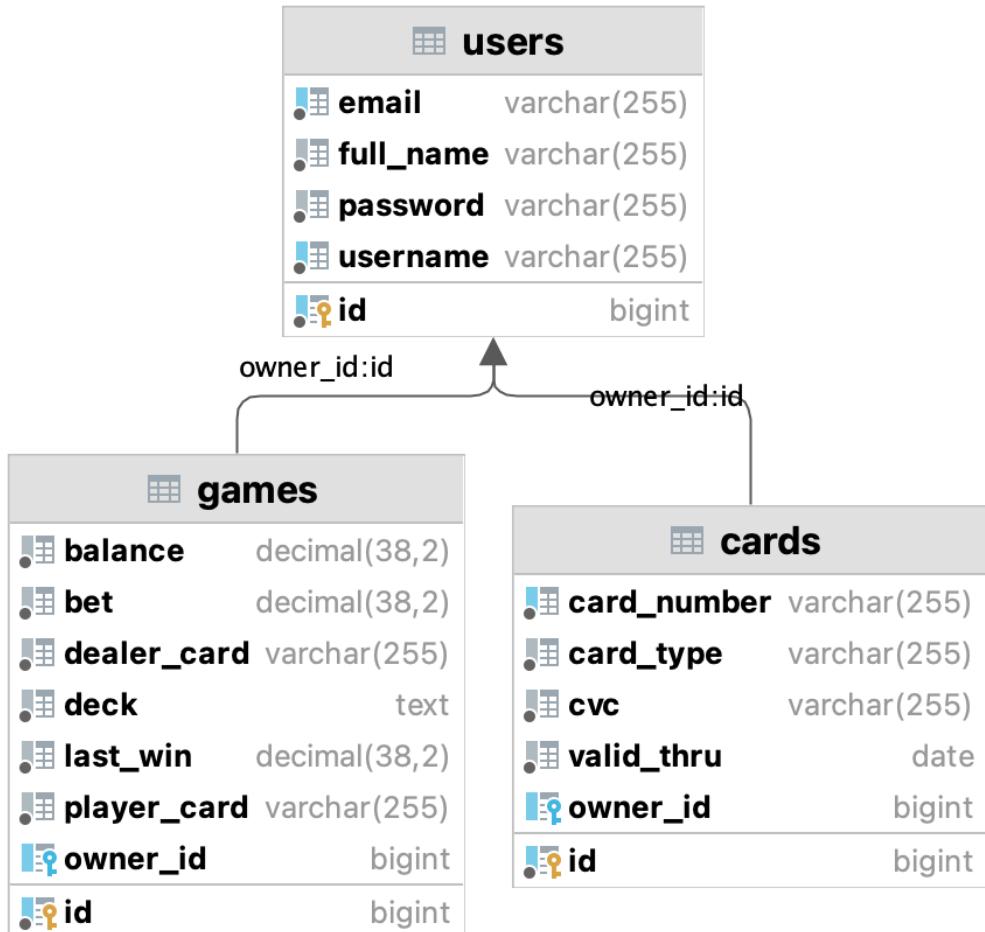
Проекта се придържа към добре познатата Model-View-Controller (**MVC**) трислойна архитектура.

- Пакетна структура





- Структура на базата



4. Стартране на сървъра

От директорията CayetanoCards на проекта отворете терминал и изпълнете последователно командите :

```
./gradlew build  
java -jar ./build/libs/GuessCard-0.0.1-SNAPSHOT.jar ./src/main/java/com/cayetano/guesscard/  
GuessCardApplication.java
```

или ако използвате **IntelliJ** среда за разработка - може просто да го build-нете и run-нете с предложения от IDE-то интерфейс.

5. Автентикация

Приложението разполага с Login и Registration страници. Преди да започне всяка игра, потребителя е необходимо или да се логне със съществуващ профил или да създаде нов посредством регистрация.

Преди да започне играта, играча е необходимо да депозира сума с която да разполага по време на играта. За тази цел чрез бутона от навигационното меню "Deposit" той трябва да въведе данни за кредитната си карта, от която ще се теглят средствата при поискване. Операцията по въвеждане на кредитната карта е еднократна и при всеки следващ депозит е необходимо само да се въвеждат CVC код-а на картата и сумата, която играча желае да депозира.

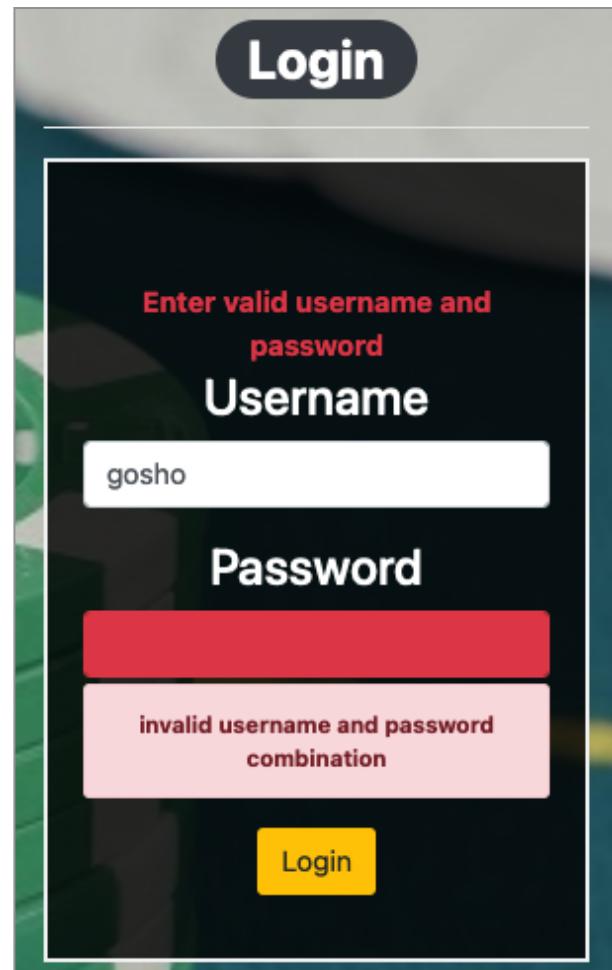
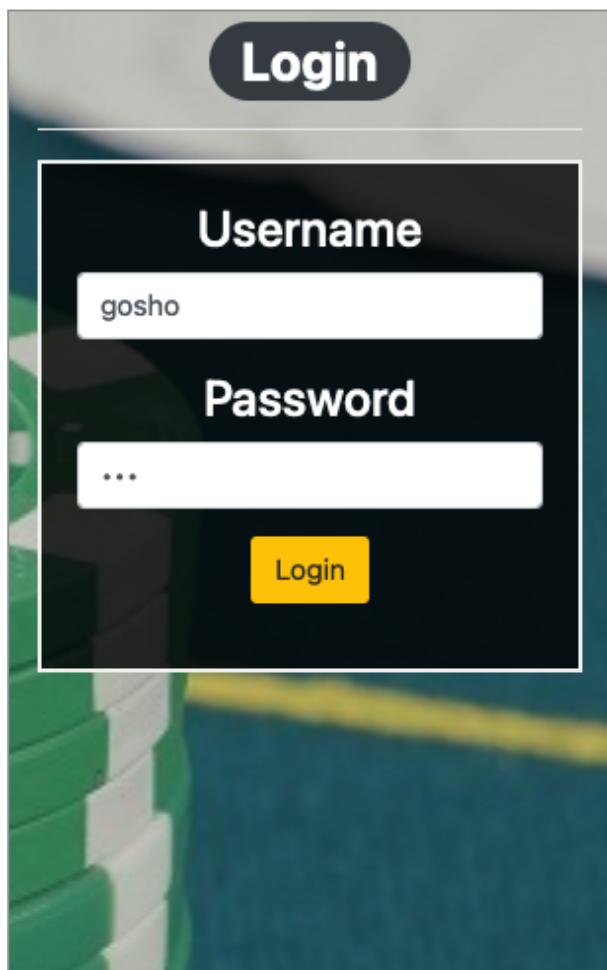
6. Валидация

Всички данни, които се въвеждат от потребителя се валидират от сървъра преди приложението да започне да работи с тях. В пакета "com/cayetano/model/validation" се намира цялата налична валидация. Тя е разделена на 3 подпакета:

- **Register:**
 - **Username:** Непразен текст с дължина от 3 до 20 символа (включително). Името трябва да съдържа поне един непразен (non-blank/non-tabulation) символ. Потребителското име се проверява за уникалност спрямо наличние в базата данни (case-insensitive).
 - **Full name:** Непразен текст с дължина от 3 до 20 символа (включително).
 - **Email:** Непразен, валиден email адрес. Email адресът се проверява за наличието на точно един символ "@". Email адресът се проверява за единственост спрямо наличние в базата данни (case-insensitive).
 - **Password:** Непразен текст с дължина от 3 до 20 символа (включително).
 - **Confirm Password:** Текст, който съвпада с текста въведен в полето Password.
- **Login:**
 - **Username:** Непразен текст с дължина от 3 до 20 символа (включително). Потребителското име се проверява за единственост спрямо наличние в базата данни.

- **Password:** Непразен текст с дължина от 3 до 20 символа (включително). Проверява се в базата данни за съществуваща комбинация от въведените потребителско име и парола (case-sensitive).
- **Add Card:**
 - **Card Number:** текст, който се валидира от регулярен израз дали е съставен от точно 16 цифри. Проверява се в базата данни за единственост.
 - **CVC:** трицифрен код на гърба на кредитната карта, който се валидира от регулярен израз дали е съставен от точно 3 цифри.
 - **Card Type:** изброен тип, който може да бъде "Master Card", "Debit Card", "Debit Card", "Credit Card" или "Disposal Revolut Card".
 - **Expired Date:** валидна дата в бъдещето.
- **Deposit:**
 - **Deposit Sum:** валидна сума във формат с нула, една или две цифри след десетичната запетая.
 - **CVC:** трицифрен код на гърба на кредитната карта съставен от точно 3 цифри. Проверява се в базата данни дали съвпада с CVC кода на въведената от потребителя карта.

При всяка невалидна въведена информация се връща подходящо съобщение на потребителя, което да му подскаже какво е объркал. Разгледайте снимките по-долу.



Register

Username
pesho

Full name
pesho

Email
asd

Password
...

Confirm Password
.....

Register

This screenshot shows a registration form with five fields: Username, Full name, Email, Password, and Confirm Password. All fields contain the value 'pesho' except for the Email field which contains 'asd'. The 'Register' button is visible at the bottom.

Register

Username
pesho

username must be unique

Full name
pesho

Email
asd

must be a well-formed email address

Password

Confirm Password

passwords mismatch

Register

This screenshot shows the same registration form as the first one, but with validation errors. The 'Username' field is highlighted in red with the error message 'username must be unique'. The 'Email' field is also highlighted in red with the error message 'must be a well-formed email address'. The 'Password' field is highlighted in red with the error message 'passwords mismatch'.

Add Card

Card Number

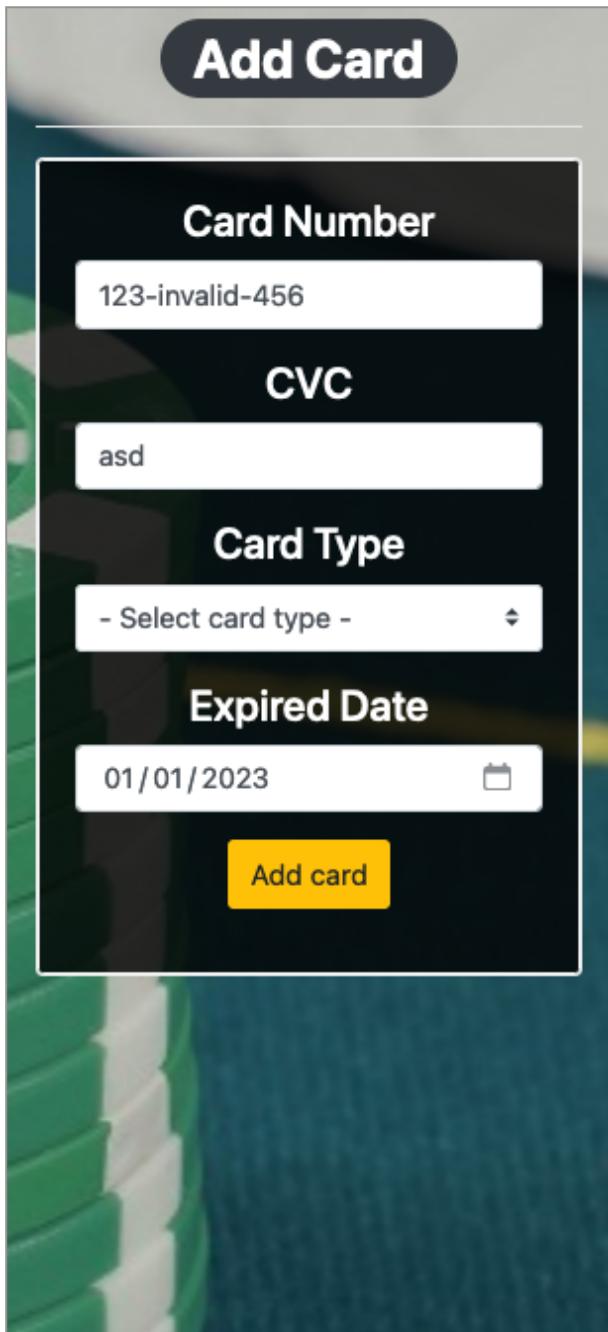
CVC

Card Type

Expired Date

Add card



Add Card

Card Number

123-invalid-456

must be exactly 16 digits

CVC

asd

must be exactly 3 digits

Card Type

- Select card type -

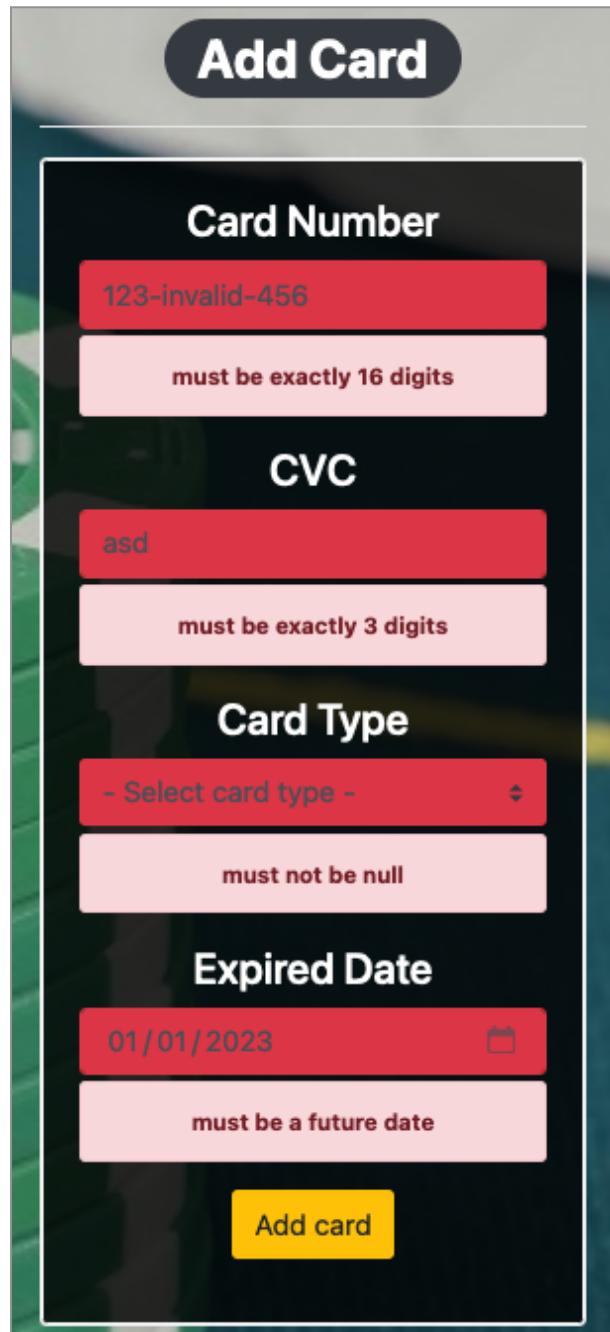
must not be null

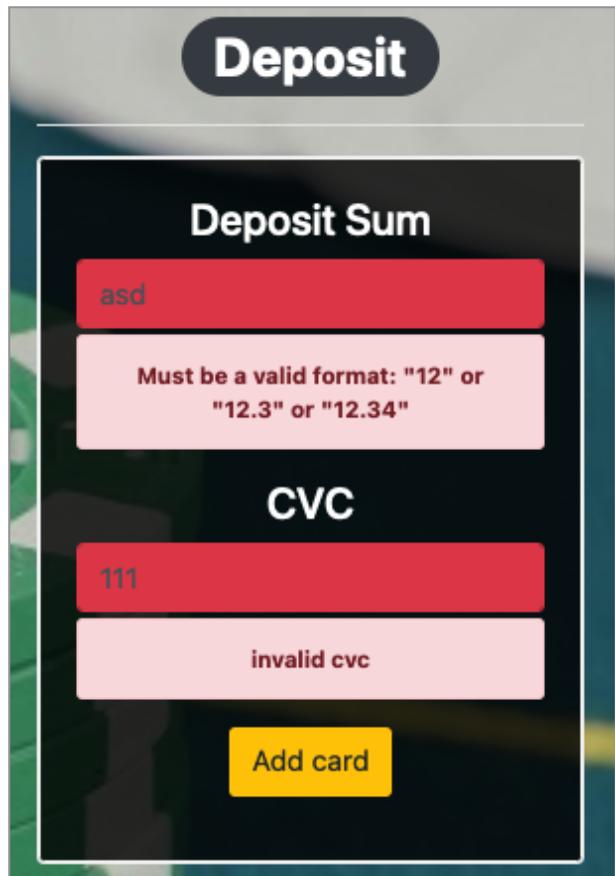
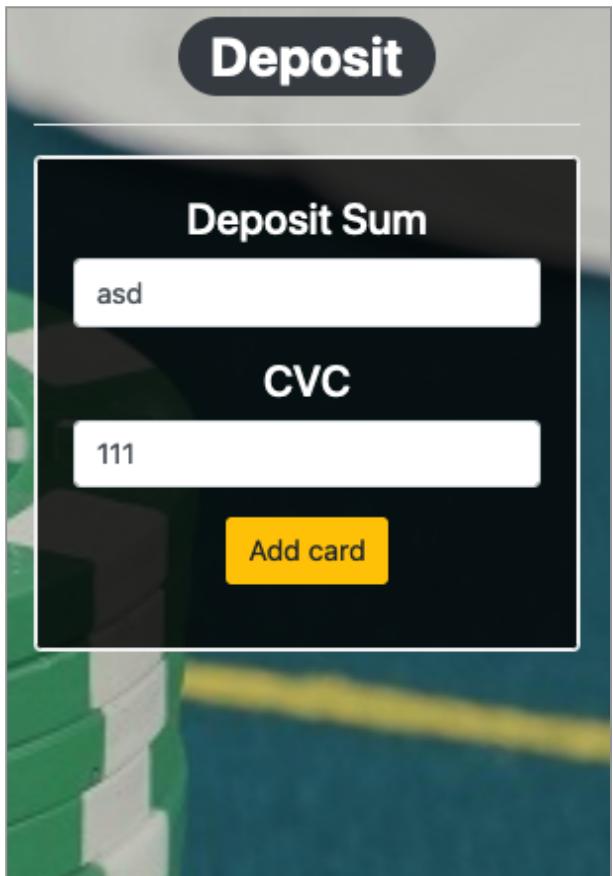
Expired Date

01/01/2023

must be a future date

Add card



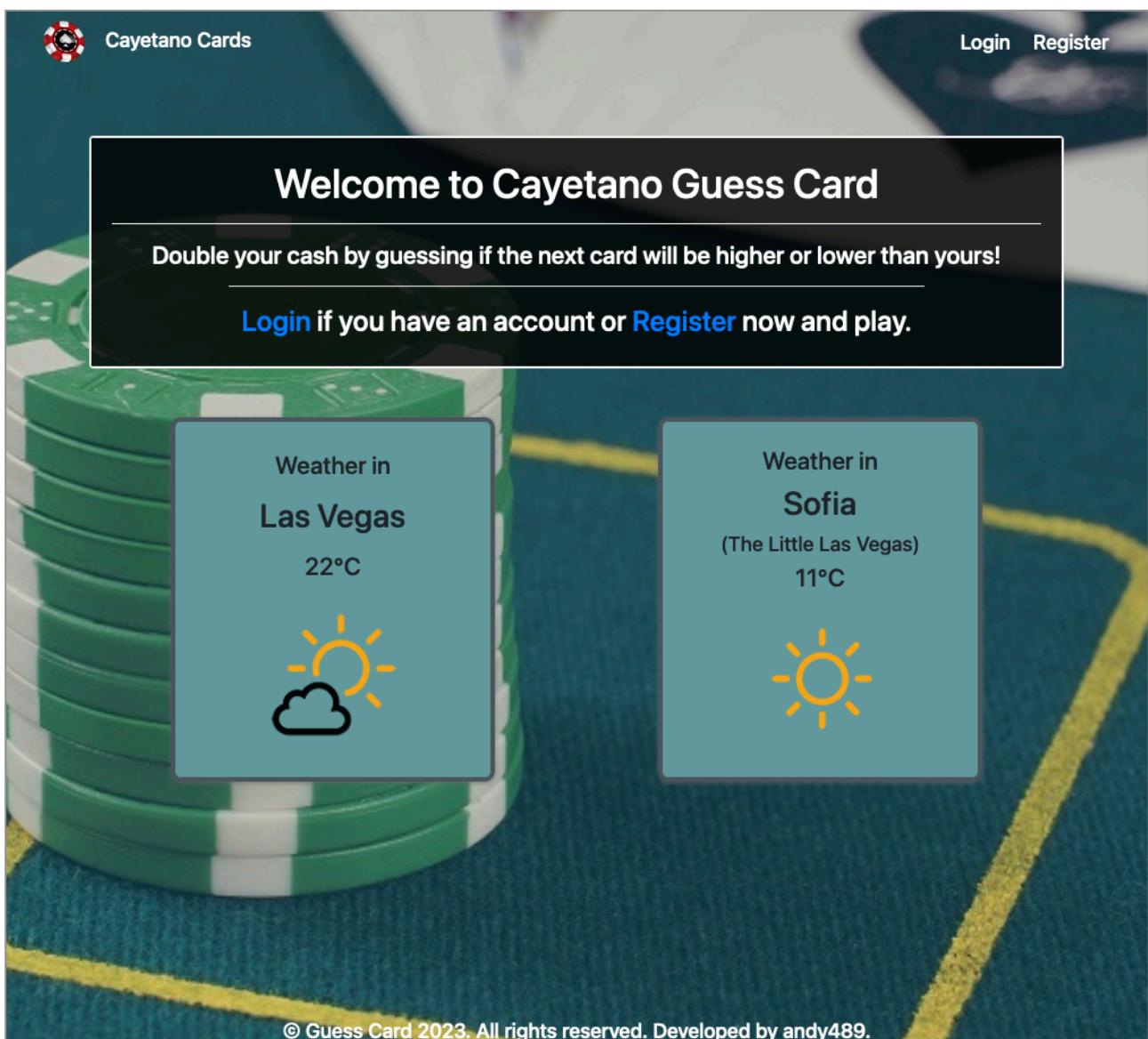


7. Нормален ход на играта

В тази точка ще проследим как ще протича обичайното потребление от потребител/играч на приложението/играта. След което, четящият този документ може да се чувства предизвикан да счупи някъде логиката като кривне от стандартния "flow".

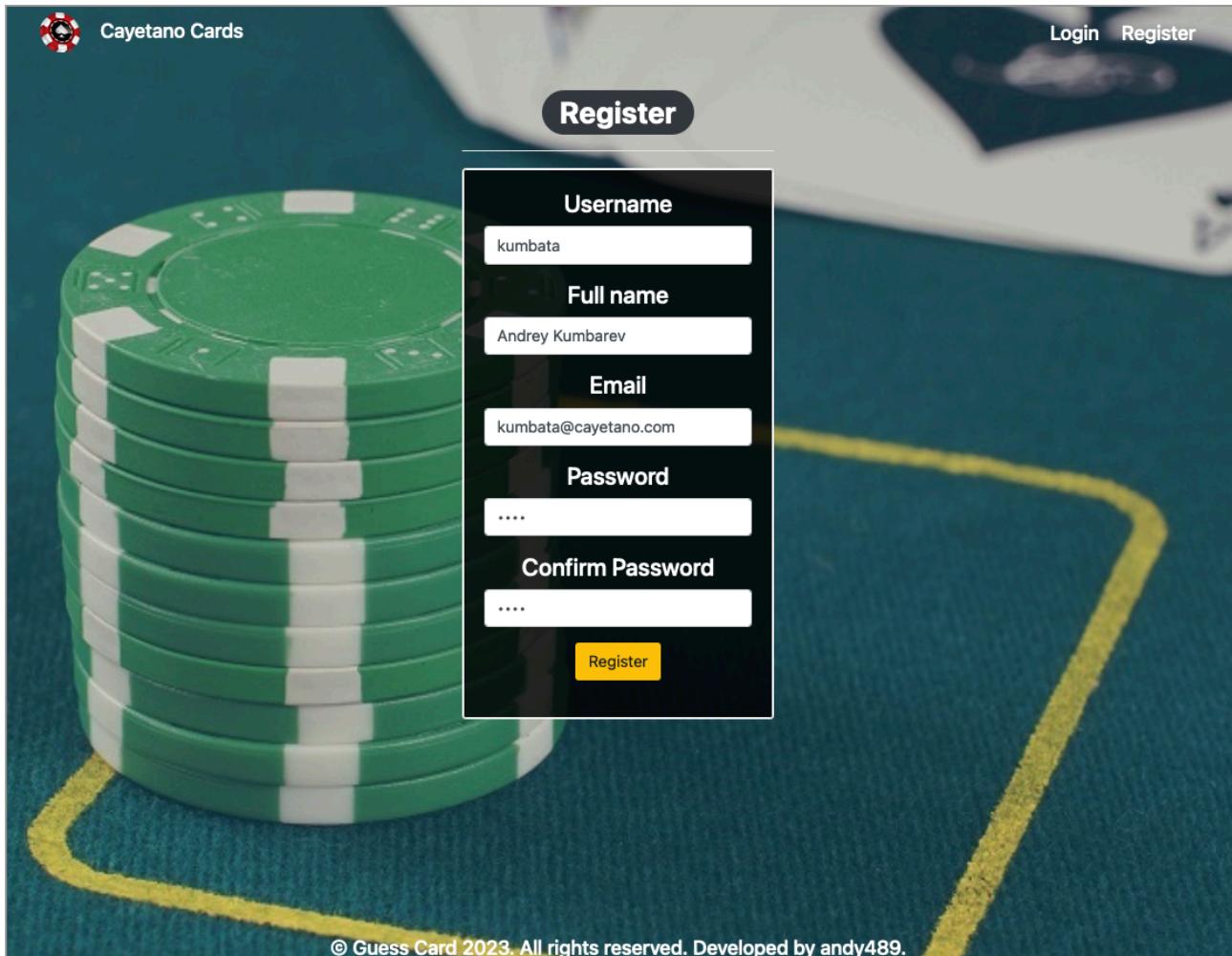
- **Начална страница на приложението**

На началната страница на приложението се предоставят на потребителя възможност за влизане със съществуваща регистрация или правене на нова такава. Използва се JavaScript Fetch API - то за визуализиране на реалното време в Лас Вегас и София. Тази визуализация може и да се направи с RestTemplate и Server-Side rendering чрез Java код, но тъй като целият сървър е написан на Java, реших да делегирам тази фасадна функционалност на JavaScript.



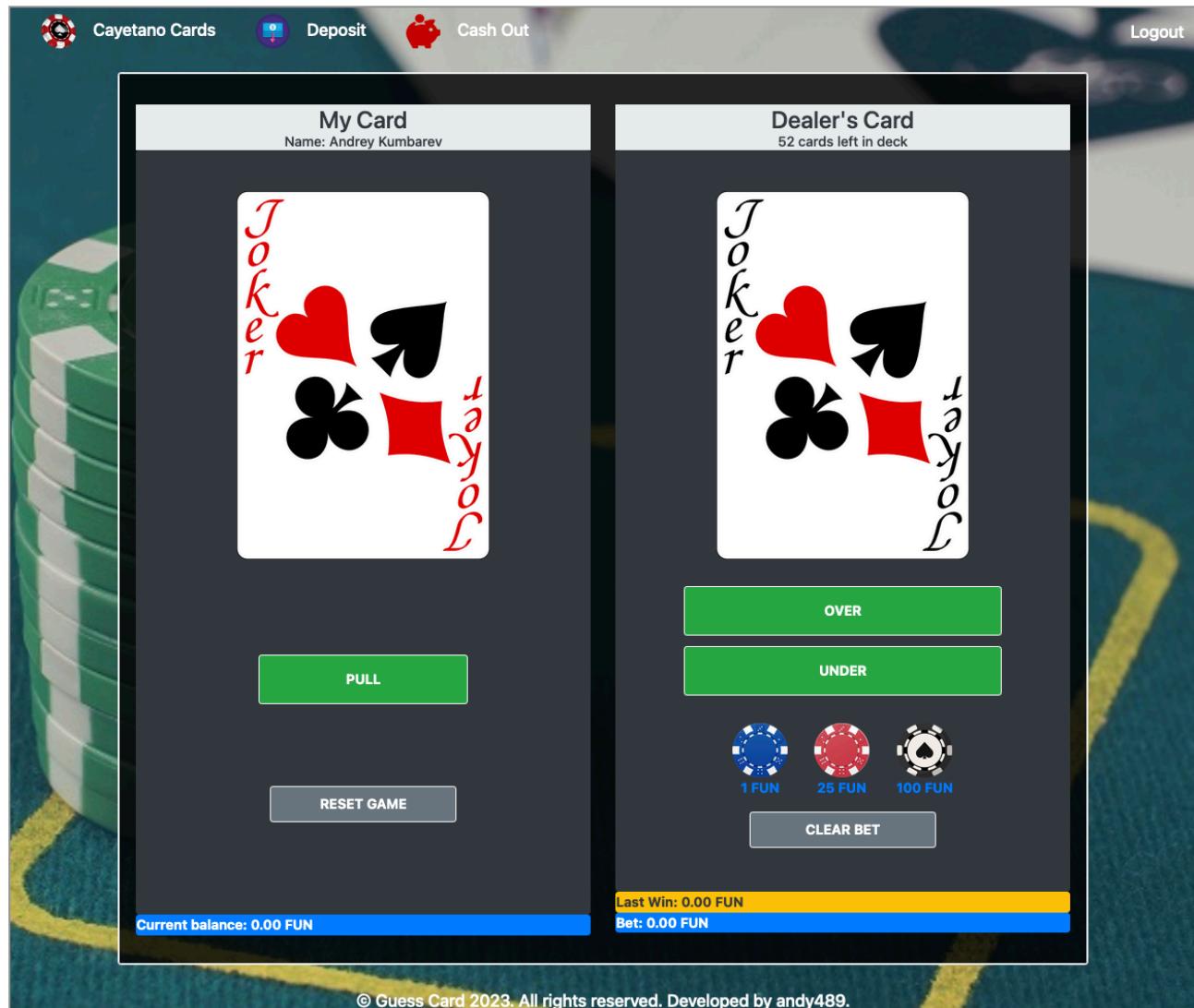
- **Регистрация**

Потребителят се регистрира в приложението като въвежда валидни данни във формата за регистрация. (Всяка прилика с реални лица може да се счита за случайна).



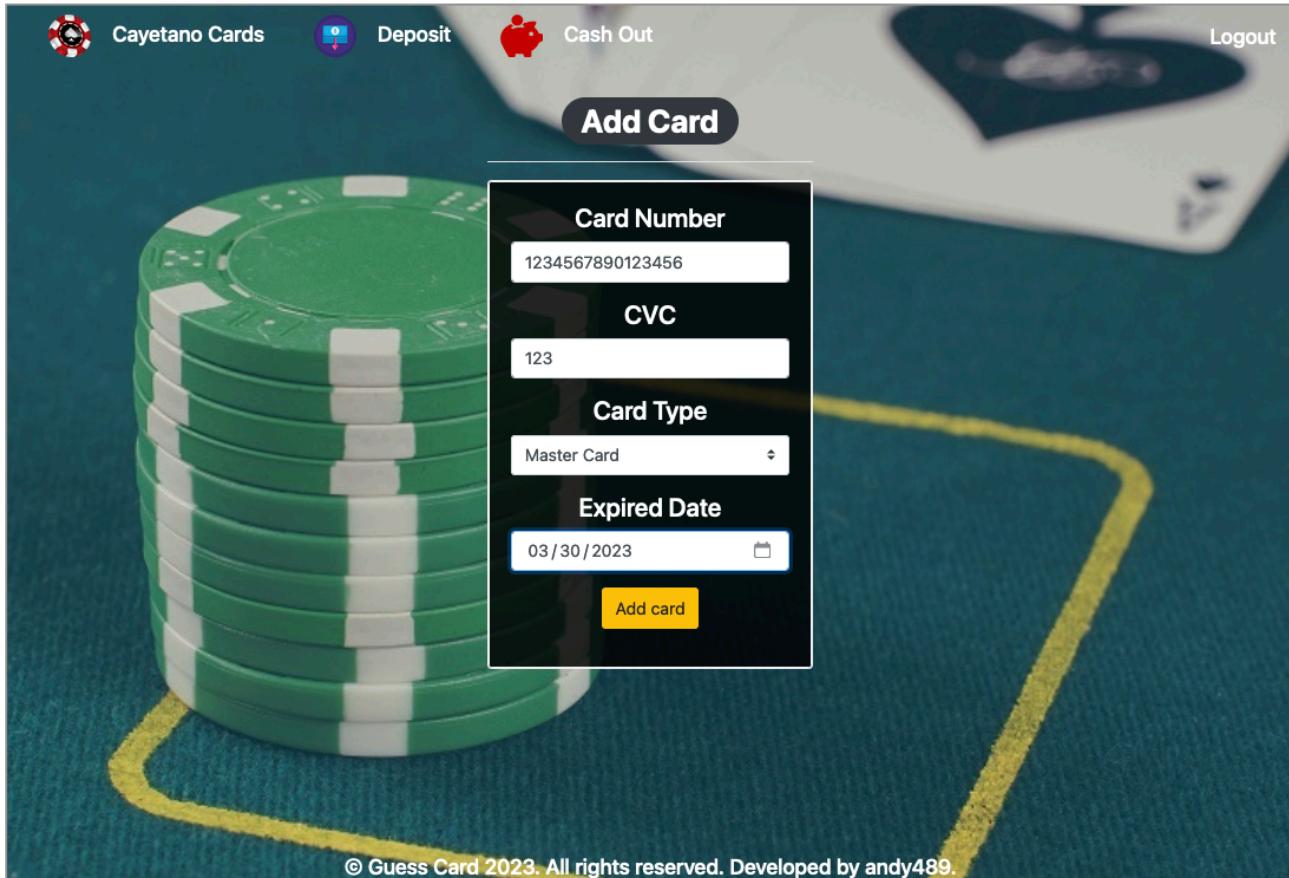
- **Home страница**

Home страницата е мястото към което се препраща потребителя след успешна регистрация или след успешен логин. Това е мястото където „живее“ играта. Първоначално картите по подразбиране и за играча и за дилъра ще са жокери, тъй като състоянието на играта е начално.



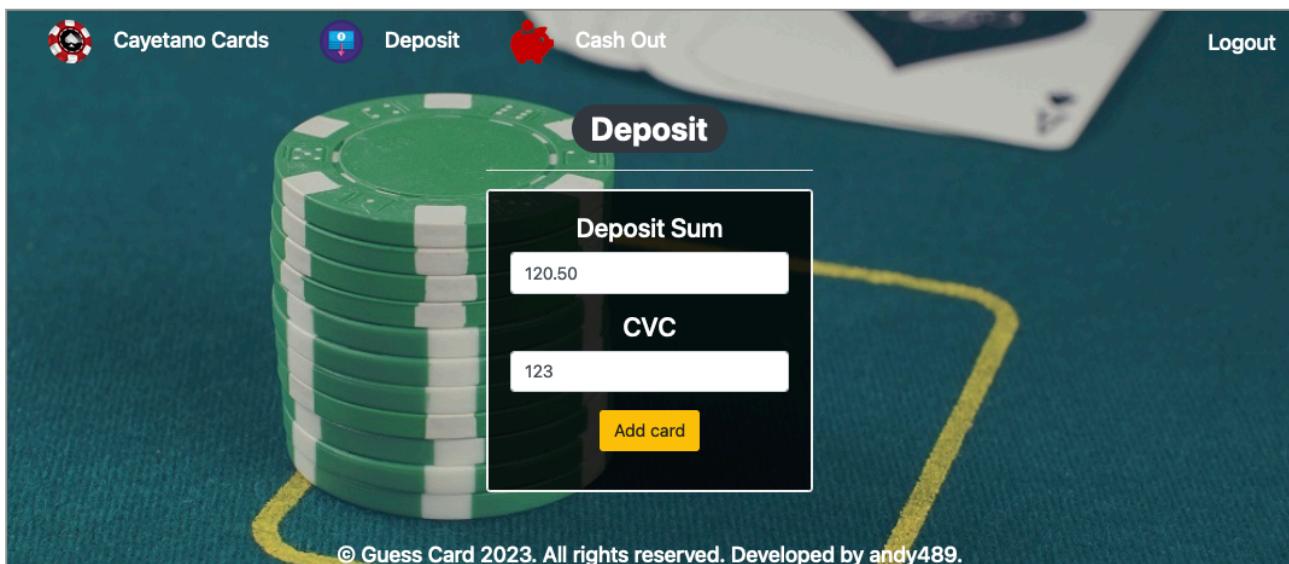
- **Добавяне на кредитна карта**

Тъй като не е имплементиран начален бонус при регистрация, първото нещо което следва да направи потребителя е да регистрира банкова карта от която да депозира средства, с които да играе. Това действие ще е еднократно, тък като при всеки следващ опит за депозит, текущият потребител ще се асоциира с вече въведената от него карта и ще е необходимо само да валидира картата със същото CVC и да заяви желаната сума за депозит (виж следващата стъпка).



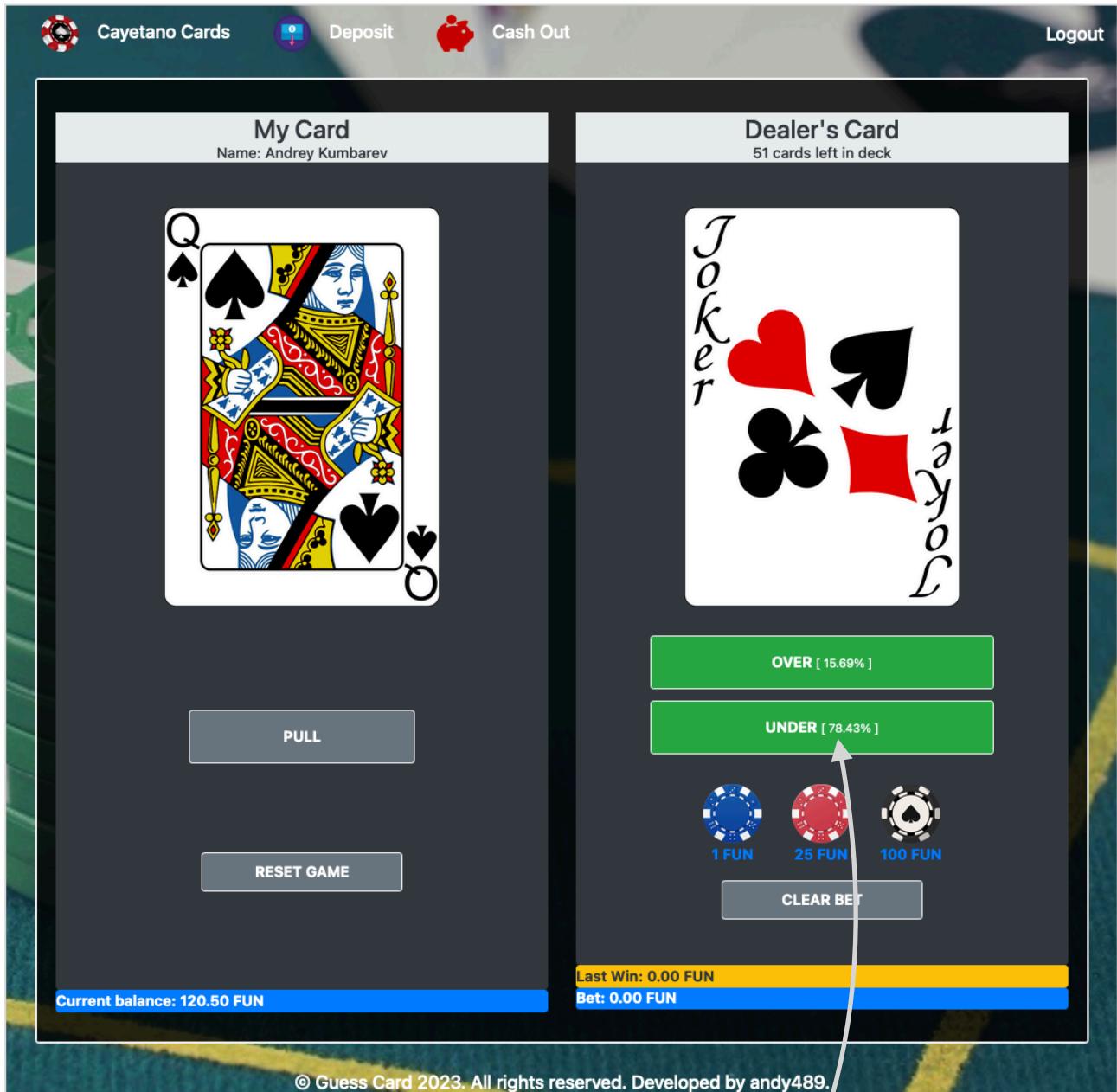
- **Депозит**

След като потребителят е регистрирал успешно разплащателна карта ще бъде препратен към страницата за въвеждане на желана сума.



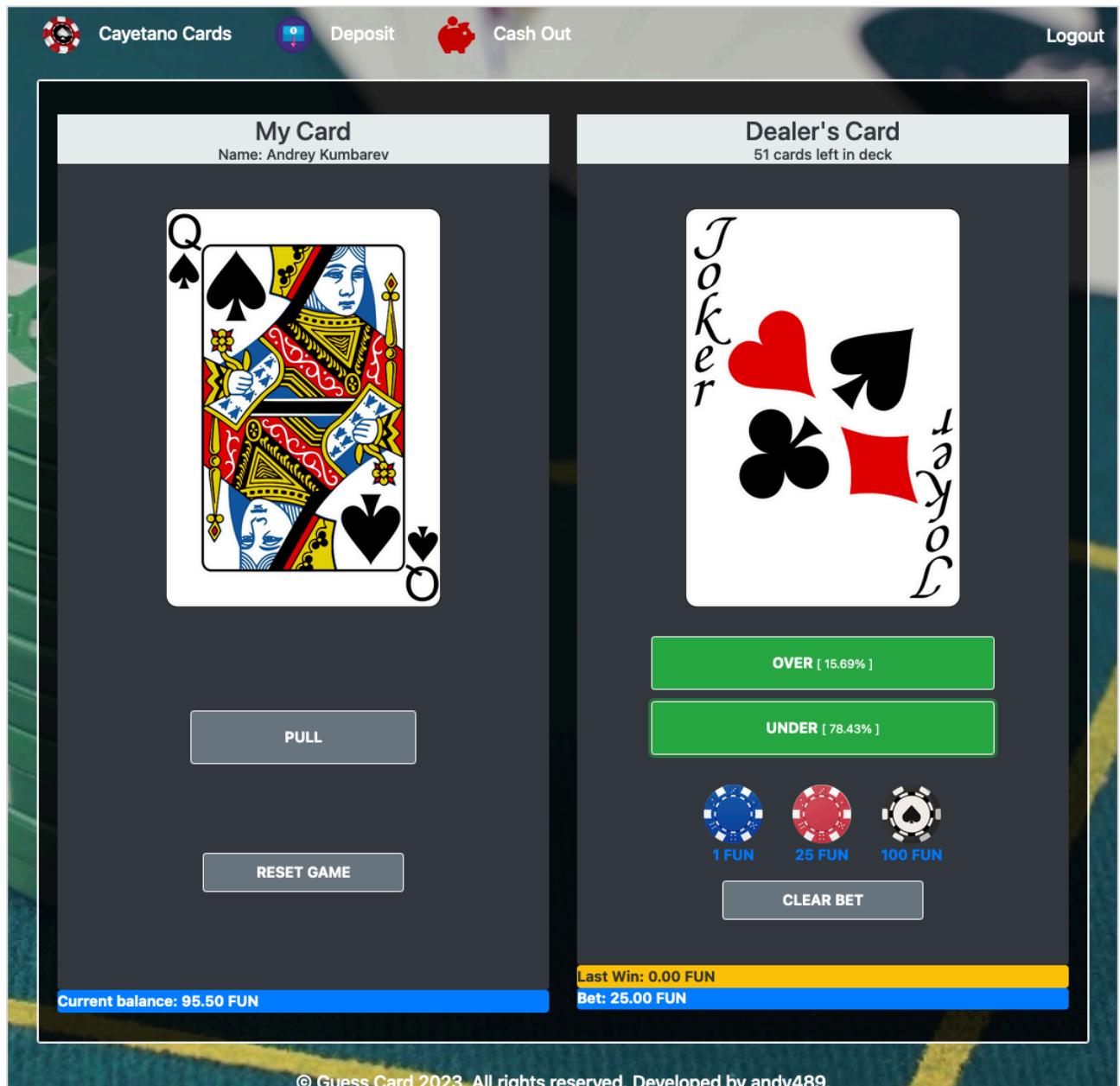
- **Изтегляне на карта за играта от началното тестве**

След изтегляне на карта, на играчът ще му се подсказва автоматично на двета бутона за залагане **OVER** и **UNDER** с процентно съдържание - на кой залог е „най-разумно“ да се играе (каква е вероятността за успех при съответния залог).



- Поставяне на залог

Играчът избира големината на залога посредством три вида чипове. При натискането на всеки чип, залогът се акумулира и баланса се актуализира. След фиксирането на сумата на залога се избира вида на залога. В случая играчът се доверява на подсказката и избира UNDER.





Cayetano Cards



Deposit



Cash Out

Logout

My Card

Name: Andrey Kumbarev



PULL

RESET GAME

Current balance: 95.50 FUN

Dealer's Card

50 cards left in deck



OVER

UNDER



1 FUN



25 FUN



100 FUN

CLEAR BET

Last Win: 0.00 FUN

Bet: 0.00 FUN

© Guess Card 2023. All rights reserved. Developed by andy489.

- Край (Logout)

7. Коментари

Лесно се забелязва, че при така формулираните правила от условието, даден играч може да си формулира винаги печеливша стратегия. За да се направи истинска казино игра е необходимо при всяка една стратегия която играчът прилага, RTP (Return To Player) коефициента да е $< 100\%$ ($\sim 97\%$). Това може да се постигне като променим правилата по следния начин:

- Да се задават коефициенти, които се калкулират спрямо вече калкулираните вероятности за успех, които предоставяме на играча като подсказки.
- Играчът да изгубва залога си в случаите когато не е познал изтеглената от дилъра карта дали ще е с по-висока или по-ниска номинална стойност И ако изтегли карта със същата номинална стойност.

С така преформулираните правила RTP-то ще е $\sim 94.12\%$ при най-добрата стратегия, която е - „винаги след първия залог тестето се рестартира (RESET GAME)“. Ако искаме да покачим RTP-то може да предефинираме изграча нито да губи нито да печели при изтегляне на карта със същата номинална стойност, ако картата е от различна боя и да губи ако картата е със същата номинална стойност и същата боя. Така RTP-то ще се повиши от $\sim 94.12\%$ до $\sim 98\%$.

Така ще получим казино игра, която в дългосрочен план ще генерира $\sim 5.88\%$ печалба от всички направени залози. Освен това, потребителят ще може да си настройва сам волатилността на играта като избира при каква начална карта какъв залог да прави. Например:

- При по-висока карта със залог OVER - по-волатилна игра.
- При по-висока карта със залог UNDER - по-ежедневна ("casual") игра.

За да работи всичко казано в коментарите е необходимо още да фиксираме залога. Тоест, веднъж изтеглил първоначална карта, играчът да няма право да променя залога до свършването на картите в тестето.