Chemistry Master

Проект № 208

Направление

Приложни програми

Автор

Денис Илиянов Цветков

Ръководител

Нели Георгиева

НПМГ „Акад. Любомир Чакалов“ – гр. София

Съдържание

Линк към GitHub repository

https://github.com/dTsvetkov13/ChemistryGame

Екип

Автор

Денис Илиянов Цветков; X клас; НПМГ „Акад. Любомир Чакалов“; e-mail: denis_tsv13@abv.bg

Ръководител

Нели Георгиева, старши преподавател по Информатика и ИТ в НПМГ „Акад. Любомир Чакалов“; e-mail: georgiewaneli@abv.bg

Консултант по дизайна

Бояна Николова, email: boyanagnikolova@gmail.com

Описание на проекта

Игра с карти свързана с химия. Тя се играе от четирима играчи, като има вариант да са или всеки за себе си, или отборно по двама.

Всеки играч започва с карти, които представляват химични елементи и карти съединения.

Целта на всеки играч е да събере, колкото се може повече точки. Точки се печелят в зависимост от това, кога си изиграл своите карти Елементи и колко точки си направил от завършени реакции. На всеки ход играчите трябва да сложат карта Елемент на масата или ако нямат подходяща, да изтеглят от тестето. Подходяща карта Елемент е тази, която съответства по група или по период на последната карта на масата.

Реакции се правят в Build Menu, до което всеки играч има достъп. За всяка правилна реакция се печелят точки. Правилна реакция е тази, за която играчът има всички карти с изходните вещества и продуктите, като след приключването ѝ тези карти се броят за изиграни.

Има два варианта на игра - Single Mode и Team Mode. В Single Mode играта приключва, когато поне двама от играчите изиграят своите карти Елементи, докато в Team Mode е достатъчно поне един от играчите да приключи.

Точкуването е следното:

- първи изиграл своите карти - 100 точки
- втори - 50 точки
- правилна реакция - 20 точки.

Целева група

Целевите групи на проекта включват:

- Ученици, които искат да подобрят знанията си по химия, по забавен и приятен начин
- Хора с интерес към химията, които да открият нов поглед към нея
- Любители на забавните и образователни игри

Основни етапи на реализирането на проекта

1. Уточняване на идеята
2. Планиране на архитектурата
3. Избор на подходящи технологии и тяхното усвояване
4. Разработване на концепция за дизайна на потребителския интерфейс
5. Реализиране на играта и тестване по време на разработката
6. Финално тестване
7. Публикуване и популяризиране на играта

Ниво на сложност на проекта

Проблеми, решени при разработката на проекта.

Backend

- База от данни - използвам Firebase Firestore, която е NoSQL. Според документацията данните трябва да са денормализирани, което спестява време при заявките и води до значително подобряване на ефективността при работа с голям обем от данни. Постигам това, като генерирам малки документи, които се пазят в колекции (виж. https://firebase.google.com/docs/firestore/data-model)
- Игрови цикъл - в него са реализирани проверки за това дали играта трябва да започне, дали е свършила и дали има нов играч, който е на ход. Когато даден играч извика `getCardDeck` или `placeCard` и резултатът от метода е положителен, записваме в базата, че играчът е приключил хода си и отразяваме тази промяна в игровия цикъл, като завършим текущата итерация и пускаме нова. Ако играчът пропусне хода си, към картите му Елементи автоматично добавяме нова от тестето.
- Осигуряване на уникалност картите на играчите - чрез въвеждане на уникален ключ за всяка карта.
- Синхронизирането на времето за изчакване на играча да извърши хода си - това осъществих чрез пускането на хронометър едновременно на клиента - за информаци на самия играч и в backend-а - за управление на игровия цикъл.
- Генериране на карти Елемент - в колекцията `elementCards` пазим предварително генерирани данни за всеки наличен елемент. Картите на играчите в базата са записани под формата на уникално `id` и име на картата, като така гарантираме, че данните за картите няма да могат да бъдат манипулирани.

Frontend

- Използвам на Material Theme за стилизирането на графичен дизайн, който лесно можем да променяме. Чрез него задавам цветовете, шрифта, форматирането и характеристиките на стандартни widget-и, от които изграждам дизайна на приложението.

Логическо и функционално описание и реализация на решението

Мобилно приложение

Всички файлове за мобилното приложение са в папка `ChemistryGame/App/chemistry_game/lib`.

- `lib/main.dart` - начален файл
- `lib/screens/wrapper.dart` - ако няма влязъл потребител, отваряме `Authenticate` екрана. В противен случай - `HomeScreen`
- `lib/screens/authenticate/authenticate.dart` - зареждаме три бутона при натискането, на които разгръщаме различни полета - форма за вход, форма за регистрация и информация
- `lib/screens/home/home.dart` - в него създаваме екран, върху който се показва екранът, избран от `Drawer` менюто. По подразбиране първо показваме `MainScreen`
- `lib/screens/home/main_screen.dart` - състои се от 2 стрелки, служещи за промяна на избрания игрови режим, поле между тях, с информация за избрания режим и бутон Play
- `lib/screens/home/friends_screen.dart` - в него показваме приятелите, за които получаваме данни от базата. Има възможност да създаваш приятелства и да виждаш поканите, които си получил

- `lib/screens/home/ranking_screen.dart` - топ 10 играчи, във вид на таблица. За промяна на различния игрови режим използваме DropDown меню, като при натискане, извикваме съответната cloud функция.
- `lib/screens/game_screens/room_screen.dart` - игралният екран.

  В горния ляв ъгъл има бутон, при натискането на който отваряме DrowDown меню с вариантите за съобщения, които играчите може да изпращат. В дясно, до него, е таймерът, в който отчитам времето, оставащо до края на хода.

  В горния десен ъгъл показваме точките на играча и бутона за напускане на играта.

  В центъра на игралното поле има 3 елемента - Deck, Last card и Build Menu. При натискане върху “Deck”, ако играчът е на ход, изтегля нова карта от тестето. Last card е последната изиграна карта. При натискането на Build Menu отваряме диалогов прозорец, в който може да се правят реакции. При натискане на бутона “+”, добавяме ново поле за карта, която поставяме чрез влачене от картите, които са в ръката ни. При натискане на бутона “тикче”, извикваме cloud функцията `completeReaction`.

  Горе, в ляво и в дясно на централното поле показваме информация за останалите играчи в стаята.

  В долната част на екрана са картите на играча, които чрез влачене може да поставяме върху полето Last card, описано по-горе. За тази цел извикваме функцията placeCard. Ако групата и периодът не съвпадат, картата не се поставя. Реализирано е странициране на картите, ако са повече от шест, като страниците се сменят със стрелки.
- `lib/screens/game_screens/summary_screen.dart` - таблица с данните за играчите след края на играта
- `lib/models/profile_data.dart` - модел с данни за играча
- `lib/models/user.dart` - модел за за аутентикацията
- `lib/models/card.dart` - абстрактен клас за карти. Съдържа полета за `uuid` и `name`.
- `lib/models/element_card.dart` - модел за представяне на карта Елемент. Класът наследява `card.dart`, като добавям полета - `uuid`, `name`, `group`,

`period`, `usedInReaction`. Реализирал съм методи за рисуване на карта под формата на Draggable и на обикновен Container.

- `lib/models/compound_card.dart` - модел за представяне на карта Съединение. Наследява класа `card.dart`. Реализирал съм методи за рисуване на картите като Draggable и като Container.
- `lib/models/player.dart` - съдържа полетата `name`, `id`, `points`, колекциите `elementCards` и `compoundCards`, както и методи за визуализация на играч.
- `lib/models/reaction.dart` - модел за представяне на реакция. Съдържа колекциите `leftSideCards` и `rightSideCards`, полетата `updated` и `exists` и `draw` метод, за рисуване на лявата страна на реакцията, бутон за добавяне, стрелка за реакция и дясна страна.
- `lib/models/fieldPlayer.dart` - модел за играчите, които са на полето.

База от данни

За базата от данни съм използвал Firebase Firestore, която е NoSQL. Първоначално планирах да използвам Google Real-time Multiplayer за реализиране на backend-а на играта, но е спрян от поддръжка от 16.09.2019 и в документацията препоръчват използването на Firebase Realtime Database. Впоследствие избрах Firebase Cloud Firestore, понеже е по-нова, по-бърза и по-интуитивна в сравнение с Realtime Database.

Структура на базата:

users (userId):
- username
- email
- singleGameWins
- teamGameWins

tokens (userId):
- token

friends (userId):
- friends

invitations (userId):
- invitations

rooms (roomId):
- freeSeats

- gameType

roomsData (roomId)

- players
- gameFinished
- subscribedTokens
- leftPlayers
- finishedPlayers
- gameType

roomCardsData (roomId):

- deck
- lastCard

roomsTurnData (roomId):

- finishedTurnPlayer
- nextTurn
- finishedPlayers
- readyPlayers

players (playerId)

- name
- points
- roomId
- compoundCards
- elementCards

elementCards (cardName)

- symbol
- group
- period

compoundCards (cardName):

- points

Firebase Cloud Functions

Cloud функциите са пишат на езика Typescript. Чрез тях реализирам backend методи, публикувани на сървърите на Google, които се изпълнявам чрез заявки или чрез Firebase triggers. Използвайки този вид архитектура, имам възможността без проблем да поддържам голям брой потребители, които да играят едновременно, тъй като от

cloud инфраструктурата на Google получавам точните ресурси, които да отговарят на текущото потребление.

Реализирал съм следните функции:

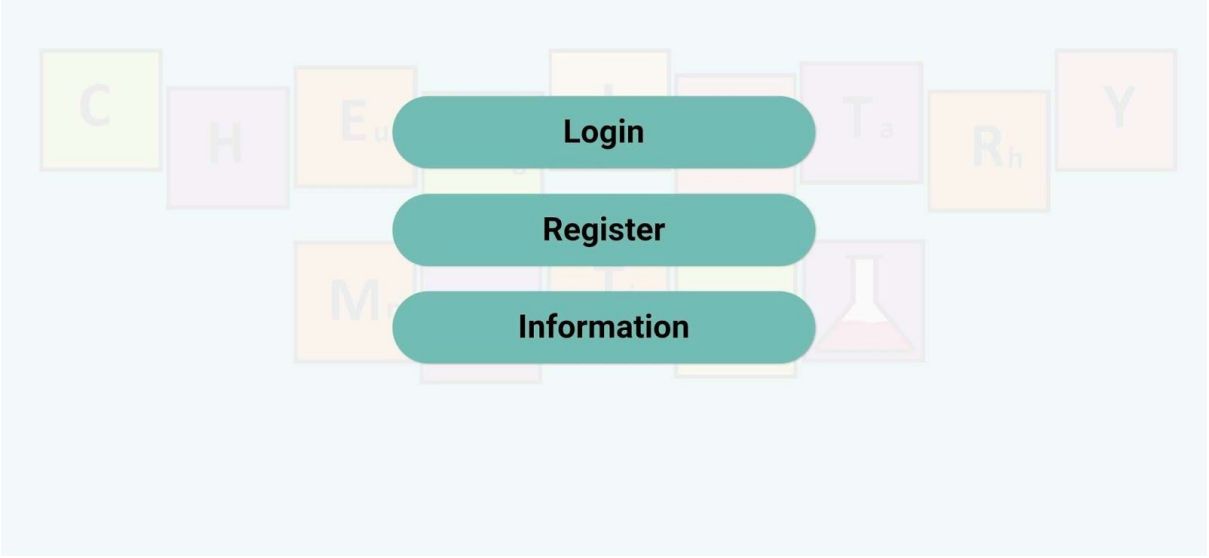
- `updateUser` - за актуализиране даден потребителски профил
- `createRoom` - за добавяне на запис в `rooms` и `roomsData` за новата стая, като преди това създаваме уникално `ID` за нея
- `updateRoom` - актуализиране дадена стая, като добавяме новите играчи в базата и намаляваме стойността на брояча в `rooms/freeSeats`
- `findRoom` - търсим стая по зададен `gameType` и ако намерим, изпълняваме `updateRoom` за стаята. В противен случай създаваме нова стая
- `listenersToRoomFreeSeats` - когато `rooms/freeSeats` стане 0, започваме игра за дадената стая, като създаваме данните за всеки играч в базата и генерираме картите. След това изпращаме съобщение, че играта е започнала
- `listenersToRoomTurnData` - тук е реализиран игровият цикъл на играта (виж Ниво на сложност на проекта -> Игрови цикъл)
- `readyPlayer` - увеличаваме с едно брояча за готовите играчи в базата
- `getPlayerCards` - връщаме картите за даден играч и неговия `username`
- `getDeckCard` - раздаваме карта от тестето на играча и записваме в `roomsTurnData`, че играчът е завършил хода си
- `placeCard` - сравняваме картата на терена с тази на играча и ако те съвпадат по група или по период, поставяме картата на играча, обновяваме базата и изпращаме съобщение за нова карта на терена
- `sendChatMsgToEveryone` - изпращаме съобщение на всички играчи в стаята
- `addLeftPlayer` - добавяме играч към тези напуснали стаята. Ако те станат повече от един, играта приключва и останалите двама печелят
- `sendTeamGameInvitation` - изпращаме покана за Team Mode игра
- `completeReaction` - правим заявка към Wolfram Alpha API за дадената реакция и анализираме отговора. Вариантите за отговор са: че реакцията е правилна, че липсва даден елемент, че реакцията не е правилна или че една от страните на реакцията е празна. В случая, когато един от елементите на реакцията липсва, добавяме съответния елемент към картите Елементи на играча, като отнемаме останалите, които е използвал в реакцията

- `getProfileData` - за данните за играча
- `addFriend` - изпращаме покана за приятелство
- `acceptInvitation` - приемаме покана за приятелство
- `declineInvitation` - отказваме покана за приятелство
- `getAllInvitations` - връщаме всички покани към дадения потребител
- `getAllFriends` - връщаме всички приятели на дадения потребител
- `getTopTen` - връщаме топ 10 играчи за избрания игрови режими
- `updateCurrToken` - обновяваме token-а на играча
- `getOnlineFriends` - връщаме всички играчи, които са на линия
- `acceptTeamInvitation` - приемаме Team Mode покана

Описание на приложението

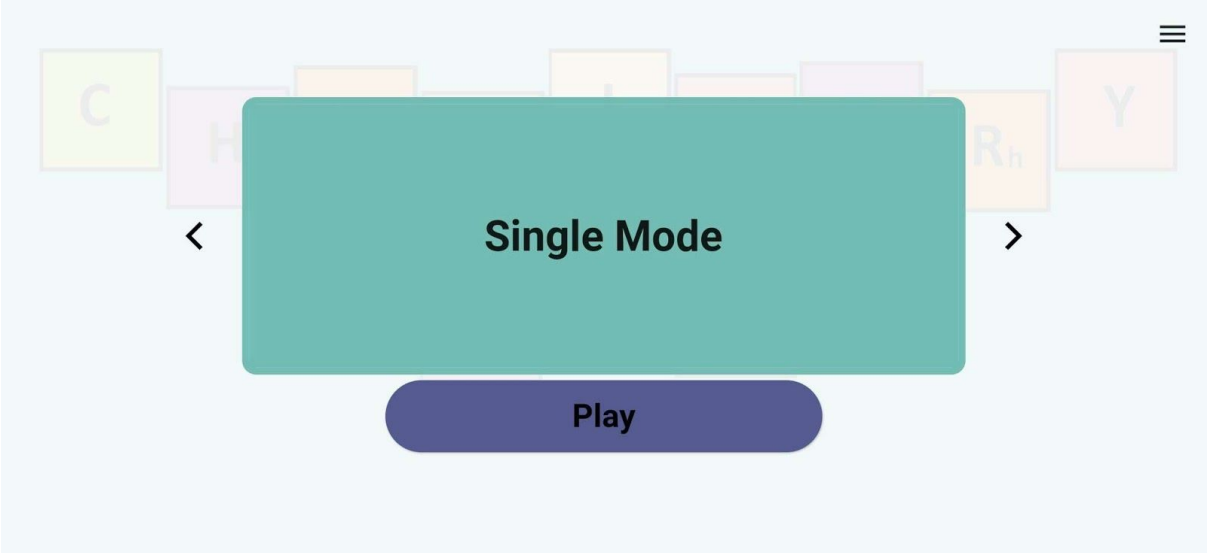
Влизане в приложението

1. Отваряме приложението
2. Регистрация/Влизане в профила

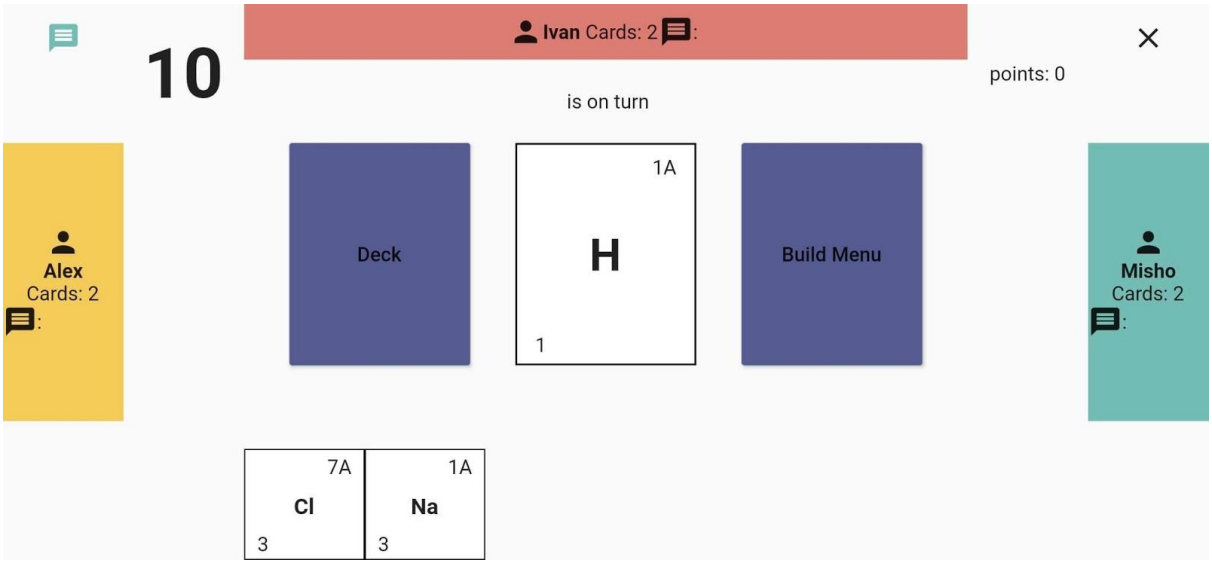


Влизане в игра

1. Избираме игрови режим и цъкаме върху бутона Play

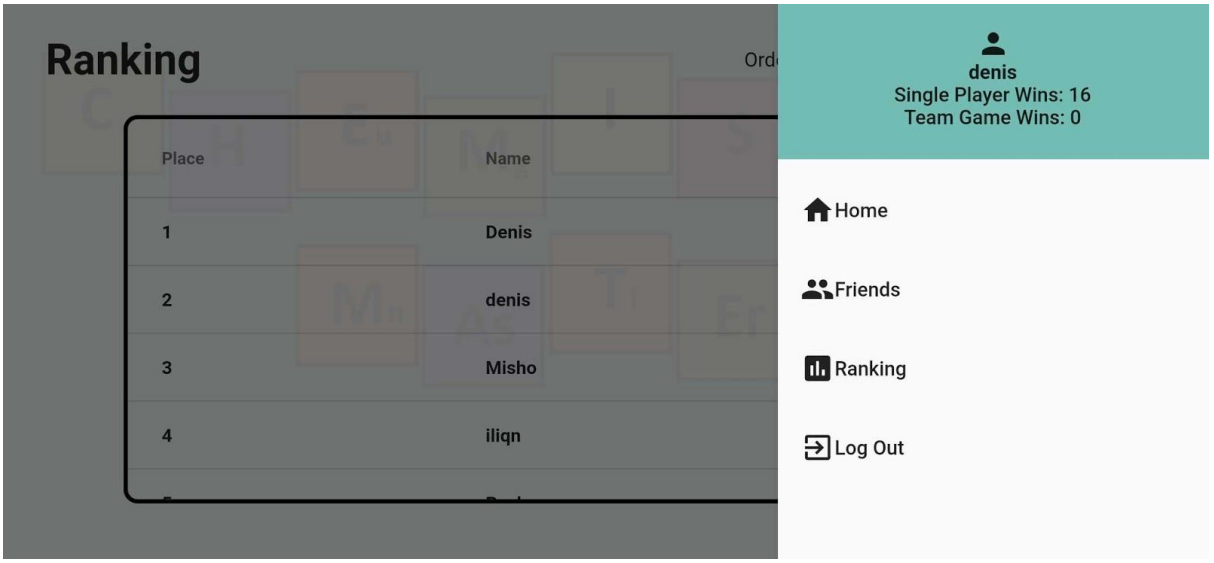


2. Изчакваме, докато се открие стая и тя се запълни
3. Започва играта

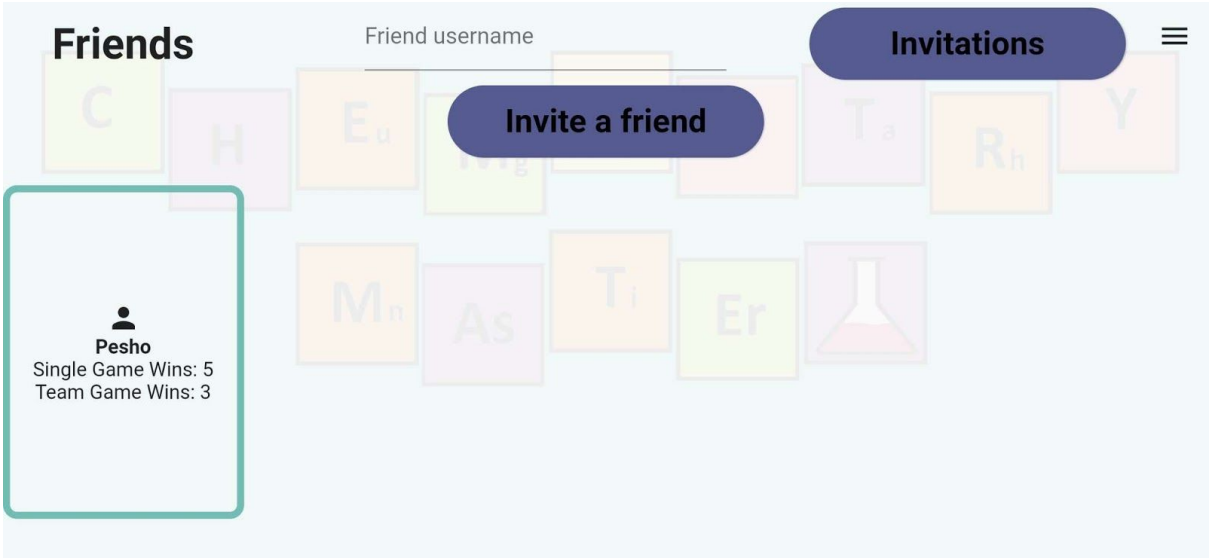


Странично меню

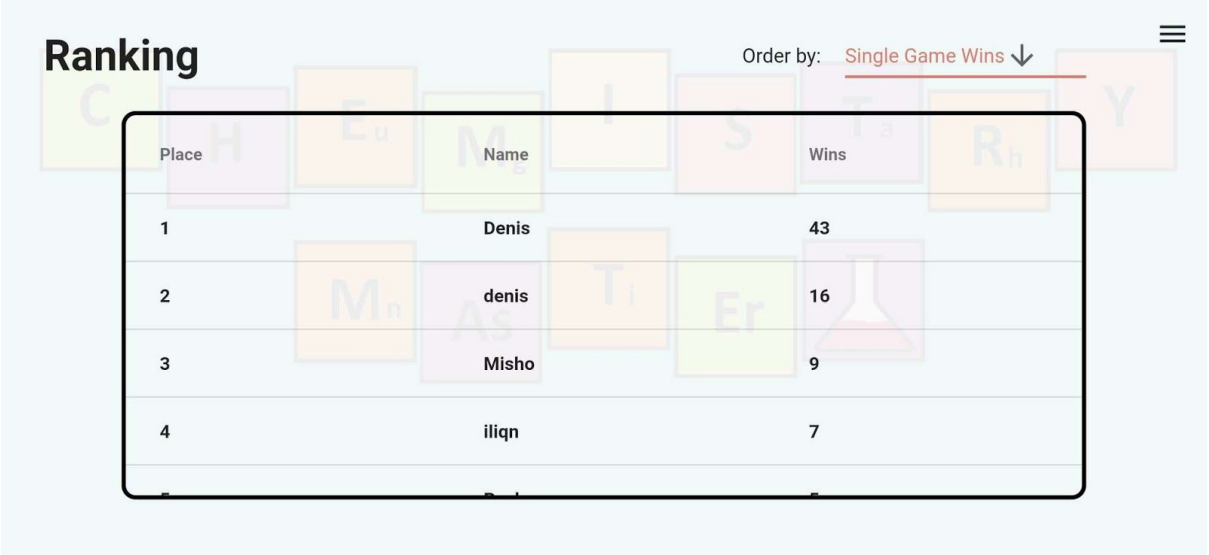
- Най-горе - данните за профила на потребителя.



- Friends - показваме приятелите на потребителя, възможност за добавяне на приятел, бутон за показване на получените поканите



- Ranking - показване на топ 10 за различните игрови режими

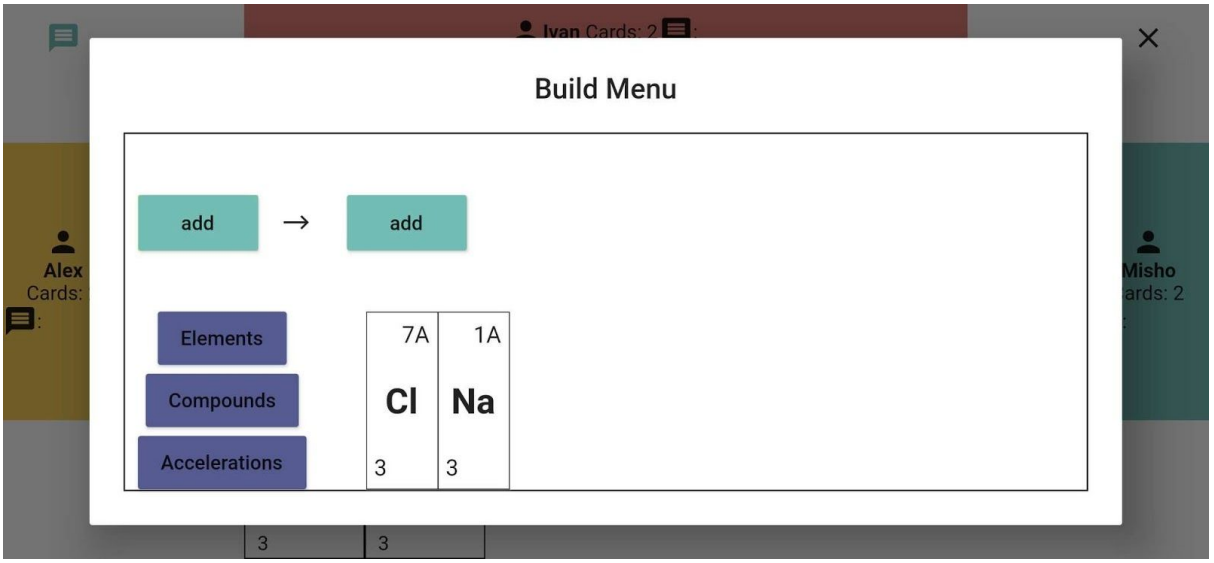


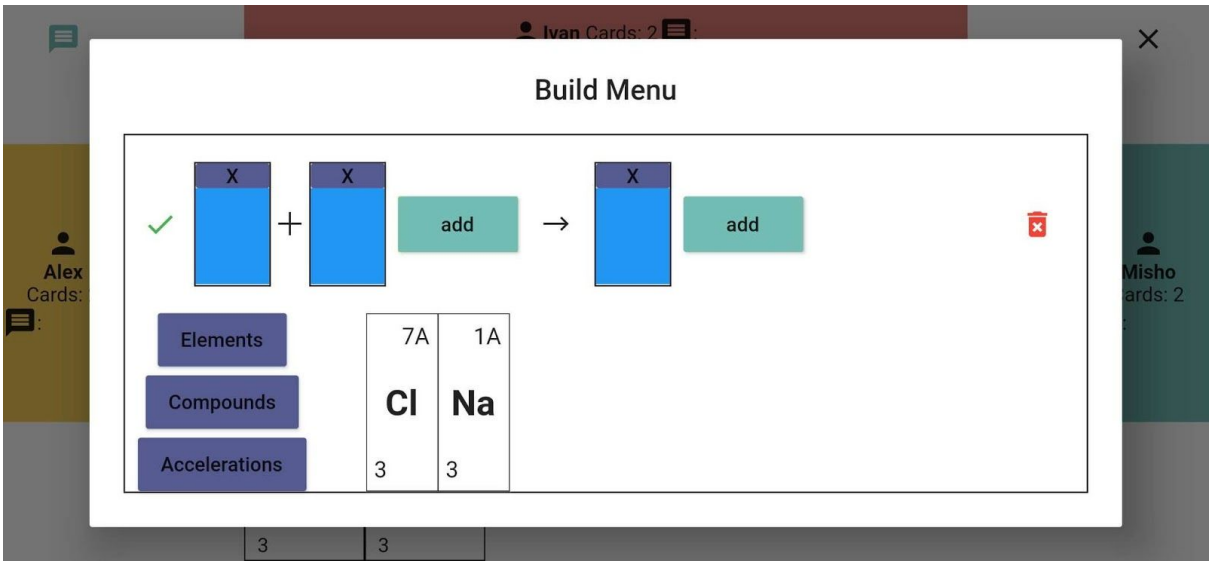
- Log out - излизаме от профила

Как се играе

- За да поставим карта Елемент на полето, избираме картата и чрез влачене я добавяме върху последната изиграна карта на полето. Това става единствено, когато ние сме на ход.
- Когато сме на ход и искаме да изтеглим карта от тестето, цъкаме върху бутона “Deck”, който се намира вляво от последната карта.
- Как правим реакция? Реакции се правят в “BuildMenu”. Отваряме го, като там са нашите карти Елементи, а над тях - бутон за добавяне на реакция. Натискаме бутона и виждаме полета за поставяне на изходни вещества или продукти. От полето “Compounds” може да видим картите Съединения, които имаме. След като изградим структурата на реакцията натискаме бутона за приемане. Ако реакция е правилна, ние ще получим точки и картите, които сме използвали, ще бъдат премахнати от тестетата ни. Ако тя е правилна, но е липсвал един продукт или едно изходно вещество, то тогава липсващите карти ще бъдат добавени към към тези в ръката ни, а използваните ще ни бъдат отнети. При неправилна реакция виждаме съответното съобщение.

  В дясно до реакцията има бутон, който позволява изтриването ѝ.





- По време на играта можем да изпращаме съобщения до другите играчи от списък с предварително генерирани такива. Това става чрез натискането на бутона в горния ляв ъгъл.
- Може и да излезем от стаята, чрез натискането на бутона в горния десен ъгъл.

Използвани технологии

Мобилно приложение

Езикът Dart с framework Flutter.

Backend

- Firebase Cloud Firestore - NoSQL база от данни
- Firebase Cloud Functions - за реализиране на backend методи
- Firebase Authentication - поддръжка на потребителски профили
- Firebase Cloud Messaging - изпращане на съобщения от cloud функциите към мобилното приложение
- Firebase Remote Config - за съхранение на текстовете, които сме използвали в приложението. По този начин може лесно да ги променяме без да е нужно да се пуска нова версия за всяка промяна.

Среди за разработка

- Android Studio - за Flutter
- Visual Studio Code - за Typescript

Заключение и идеи за бъдещо развитие

Играта Chemistry Master е забавен и приятен начин да подобрите знанията си в областта на химията, като освен това предлага различен поглед към предмета и науката като цяло.

Структурата на играта дава възможност за лесно и удобно разширение. Планирам добавянето на нови игрови режими, като Single Player Mode, при който играчът ще надгражда съединенията си - например, започвайки от основа и киселина, ще получи сол и вода, който след това ще използва за получаването на нови съединения.

Също ще бъдат добавени постижения, които играчите ще могат да постигат, както и задачи, които ще могат да приемат, когато пожелаят и съответно ще трябва да изпълнят.

Очаквам, чрез играта потребителите да увеличат своя интерес към химията и най-вече да се забавляват, докато играят.