**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Сайт за игра на шах

Дипломант: Научен ръководител:

*Благовест Атанасов Таня Петрова от Astea Solutions*

СОФИЯ

2 0 2 1

**Chess King - WEB приложение за игра на шах**

**УВОД**

Шахът – една от най-известните логически игри и единствената настолна игра, която се зачита за спорт. Шахът винаги е бил модерен, като правилата му не са сложни, но изисква много мислене и правилно преценени ходове.

Решението да бъде имплементирана игра на шах бе взето поради заинтригуващата идея да бъде измислена реализацията на тази логическа главоблъсканица във формата на код. Играта е толкова популярна, че има голям набор от разработени и публични имплементации, които са предпоставка за намаляване на значимостта на идеята. Решено бе обаче да се направи изцяло нова реализация на проекта, без да бъде взет предвид никакъв готов код или алгоритми, а с цената на повече време всичко да бъде разработено по персонализиран начин. Имплементираната игра представлява ядрото на дипломната работа, и макар идеята да не е нова, реализацията е оригинална, което превръща проекта в интересен продукт, който също така представлява ценен опит за този, който го е разработил.

**ПЪРВА ГЛАВА**

**МЕТОДИ И ТЕХНОЛОГИИ ЗА РЕАЛИЗИРАНЕ НА WEB ПРИЛОЖЕНИЯ**

* 1. **Основни принципи, технологии и развойни среди за реализиране на WEB приложения**

WEB приложенията представляват софтуер, в който са съчетани две части – сървър и клиент.

Сървърът представлява мястото, където се съхраняват, изпращат и извикват данни, като неговата функционалност позволява на клиентите да разполагат с конкретни услуги.

Страната на клиента представлява тази част от софтуера, която е ориентирана към крайния потребител. От тук се извикват заявки към сървъра, като получените данни биват обработени и показани на потребителя по възможно най-удобния и практичен начин.

Сървърът най-често се състои от две части – приложение и база данни. Приложението може да бъде написано на голям набор от езици за програмиране, като най-използваните са *PHP*, *Python* и *Java* [1, 2, 3]. Базата данни може да бъде създадена с различни СУБД, като най-известните са *MySQL*, *SQL Server* и *PostgreSQL* [4, 5, 6]. Базата данни съхранява информацията, която е нужна под формата на таблици, а чрез приложението тази информация бива създавана, редактирана и извиквана от клиента за по-нататъшна обработка.

Страната на клиента представлява WEB частта на приложението, на практика – уебсайта. Той е изграден чрез комбинация от *HTML*, *CSS* и *JavaScript* [7, 8, 9], като *HTML* представлява структурата на страницата, *CSS* оформя дизайна на елементите на базово ниво, а *JavaScript* е езика, който обработва заявките към сървъра и прави страницата функционална и представителна за потребителя.

Съществуват много библиотеки и разширения както за сървърната, така и за клиентската част от приложението. Отдолу са показани най-известните от тях.

За сървърната част:

* *PHP*: *Laravel*, *CodeIgniter* [10, 11]
* *Python*: *Django*, *Flask* [12, 13]
* *Java*: *Spring*, *Struts* [14, 15]

За клиентската част (*JavaScript*):

* *Angular* [16]
* *React* [17]
* *Vue* [18]
* *jQuery* [19]

Използвайки библиотеки и разширения програмистите улесняват и забързват процеса си на работа, защото с по-малко написан код проектите могат да бъдат развивани до по-голяма степен на сложност много по-ефективно.

* 1. **Съществуващи решения и реализации – „chess.com“ и „lichess.org“**

****

****

**Фиг. 1.1.** Скрийншот от началната страница на сайта **chess.com**

**Фиг. 1.2.** Скрийншот от началната страница на сайта **lichess.org**

Когато говорим за WEB приложение за игра на шах, съществуват много реализации на такава платформа, като в случая обект на внимание са двете най-използвани – “chess.com” [20] (Фиг. 1.1.) и “lichess.org” [21] (Фиг. 1.2.). Двете платформи имат много сходни характеристики и функционалности, но все пак имат и някои определени разлики.

Сайтът “chess.com” е създаден преди цели 26 години през 1995г., докато “lichess.org” е открит в близката 2010г. Поради по-продължителния си живот “chess.com” е много по-известен и броя на игрите, които се играят в реално време е около 200 000-300 000, докато при “lichess.org” е в пъти по-малко – към 30 000.

Двете платформи споделят следните функционалности:

* запазване на игри на компютъра чрез специално файлово разширение, както и обратния процес - зареждането им в сайта от хранилището на устройството
* турнири
* различни настройки за времето, за което се изпълнява ход в игра
* 8 различни вида игра на шах
* игра срещу компютър
* компютърен треньор, работещ чрез решаване на пъзели и рейтинг на развитието на потребителя
* голям набор от обучителни видеа

В общите функционалности “chess.com” превъзхожда с малка разлика “lichess.org” във частта с обучителните видеа, но за сметка на това “lichess.org” може да излъчва игра на живо за специално поканени потребители, а “chess.com” няма такава опция.

Ако трябва да изберем един от двата, решението зависи изцяло от предпочитанията на конкретния потребител. В безплатната версия на “chess.com” не са включени всички функционалности, като сайта предлага три опции за платен абонамент – златен, платинен и диамантен. Конкурентната платформа не предлага същото качество на някои от функционалностите, но всичко в нея е напълно безплатно и няма никакви реклами, което я прави много привлекателна опция.

**ВТОРА ГЛАВА**

**ПРОЕКТИРАНЕ НА СТРУКТУРАТА НА WEB ПРИЛОЖЕНИЕ ЗА ИГРА НА ШАХ**

**2.1. Функционални изисквания към WEB приложение за игра на шах**

- Поддръжка на потребители

- Игра на шах в мултиплеър

- История на изиграните игри

- Рейтинг система на играчите

- Поддръжка на турнири

- Чат между играчите по време на игра

Обобщение: приложението трябва да представлява сайт, в който да се регистрират и да влизат в профила си потребители, които да могат да играят шах един срещу друг. Потребителите трябва да имат комуникация помежду си под формата на чат по време на играта. Нужна е и по-сложна форма на игра, а именно чрез формáта на турнир. Трябва да съществува рейтинг за всеки играч и история на изиграните му игри, която да позволява преглеждането на изиграна игра отново и отново.

**2.2. Съображения за избор на програмни средства и развойната среда**

Технологиите, които бяха избрани за дипломната работа са следните: *Python*, *Flask*, *SQLite* [22], *DB Browser for SQLite* [23], *SQLAlchemy* [24], *HTML*, *CSS*, *JavaScript*, *Bootstrap* [25].

*Python* е един от най-популярните, развити и функционални програмни езици. Този език е избран поради красивия му интерфейс, добрата интеграция с всякакви технологии и бързината, с която се развиват проекти с него поради високото му ниво на имплементация.

*Flask* е една от основните библиотеки на *Python* за изграждане на WEB приложения, като тя бе избрана поради факта, че беше преподавана в програмата на учебен предмет и съответно принципите й на работа са по-познати от тези на подобни библиотеки.

*SQLite* е СУБД, която интегрирана със структурите на *SQLAlchemy* представлява много лесна за използване и изключително функционална база данни, която може да бъде управлявана без чист *SQL* скрипт. *SQLite* е една от най-леките и лесни за използване СУБД и по тази причина бе избрана за проекта. *DB Browser for SQLite* бе избрана като помощна програма за преглеждане и редактиране на записите в базата данни, като тези операции се изпълняват изключително чрез нея. *SQLAlchemy* е избрана като технология поради функционалните си структури, които позволяват развитието на базата данни да се ускори значително и също така увеличава логическата връзка между базата данни и останалите части на проекта.

Комбинацията на *HTML*, *CSS* и *JavaScript* е абсолютно необходима, когато трябва да се изгради динамичен и функционален сайт към приложението. *HTML* и *CSS* са единствения вариант за базовата част на клиентската страна на WEB приложение, така че те са избрани поради нужда. *Bootstrap* е използван за много малка част от дизайна на страниците, поради конкретен елемент, вписващ се добре в цялостната картина. *JavaScript* е избран поради вече натрупан опит с технологията, а и поради лесната интеграция с *Flask*.

Използваната развойната среда е съставена от две части: *Visual Studio Code* [26] и *Ubuntu WSL 2* [27].

*Visual Studio Code* е използван като редактор за кода, защото като програма е лека, има много функционалности, които са лесно достъпни, a интерфейса е богат.

*Ubuntu WSL 2*, който се използва на *Windows 10*, представлява алтернатива на вградения в операционната система *Linux Ubuntu* терминал, като има същите функционалности и улеснява работата с *Flask* и *Git* [28], както и цялостното управление на компютърната система. Поради сложността на инсталиране и конфигуриране на операционна система *Linux*, нуждата от терминал не оправдава тази операция и затова е избран по-лесния вариант чрез интеграция с *WSL 2*, която се инсталира лесно и работи достатъчно добре като заместител.

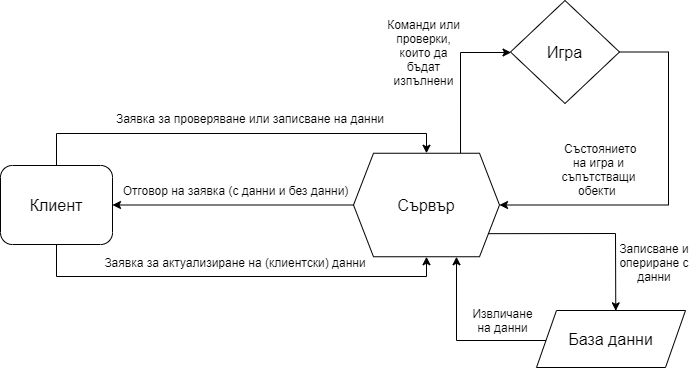
Когато се разработва проект в голям мащаб, нуждата от добре сегментиран процес на развитие на проекта, както и от гъвкаво и функционално хранилище е много важна. Затова са избрани *Git* и *Github* [29], тъй като популярността им е голяма.

**2.3. Файлова структура на проекта**

**├── auth.py ├── database.py ├── game\_base.py ├── init.py ├── main.py ├── models.py ├── multiplayer.py ├── python\_game │   ├── board.py │   ├── figure.py │   ├── game.py │   ├── player.py │   └── test\_game.py ├── static │   ├── CSS │   ├── JavaScript │   └── bootstrap-4.5.3-dist ├── templates └── tournament.py**

**Фиг. 2.1.** Файлова структура на проекта

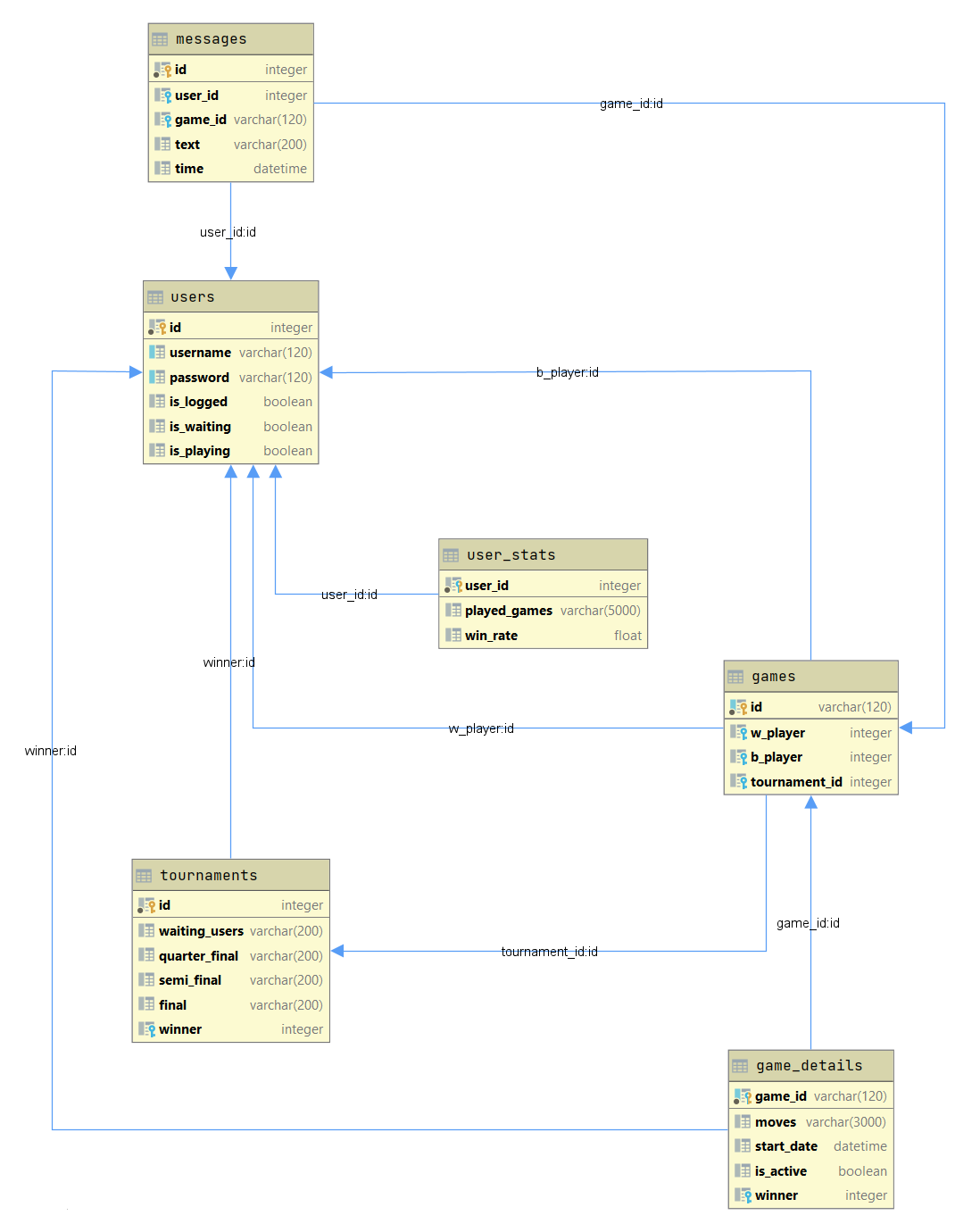
В папката „python\_game“ (Фиг. 2.1.) се намират файловете, представляващи самата игра на шах, в папката „static“ се намират *CSS*, *JavaScript* и *Bootstrap* файловете, разделени в папки, а в „templates“ са *HTML* файловете.

**2.4. Основна структура и принцип на действие на WEB приложение за игра на шах**

**Фиг. 2.2.** Структура и принцип на действие на проекта

WEB приложението се състои от четири основни части:

* Клиент – при него се показва интерфейса на сайта и играта; свързан е директно единствено със сървъра
* Игра – представлява функционалната игра на шах, от която се записват данни директно в сървъра
* База данни – оперира с информация, изпратена от сървъра
* Сървър – контролен център на приложението; след получаване на заявка от клиента той комуникира с играта и базата данни и връща отговор

**2.5. Структура на базата от данни**

**Фиг. 2.3.** Структура на базата данни

**ТРЕТА ГЛАВА**

**ПРОГРАМНА РЕАЛИЗАЦИЯ НА WEB ПРИЛОЖЕНИЕ ЗА ИГРА НА ШАХ**

**3.1. Ядрото на проекта – играта на шах**

Играта на шах е имплементирана под формата на програма от команден тип, написана на *Python*, която се базира на ООП логиката за разделяне на отделните компоненти на програмата в класове.

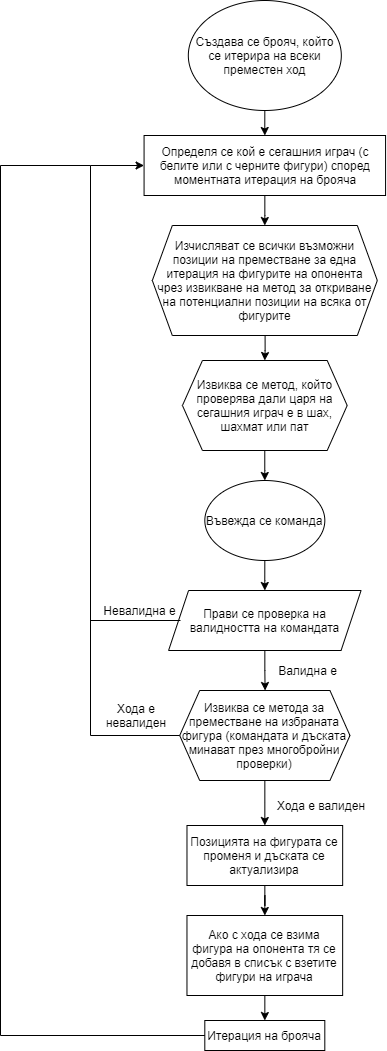
Играта се състои от пет класа:

* *ChessBoard –* дъската в играта
* *Figure* – фигурата на даден играч
* *Player* – играч
* *Game* – събирателен клас, който изпълнява цялостния процес, наречен „игра“
* *TestGame* – *unit*-тестове, които проверяват правилното функциониране на играта

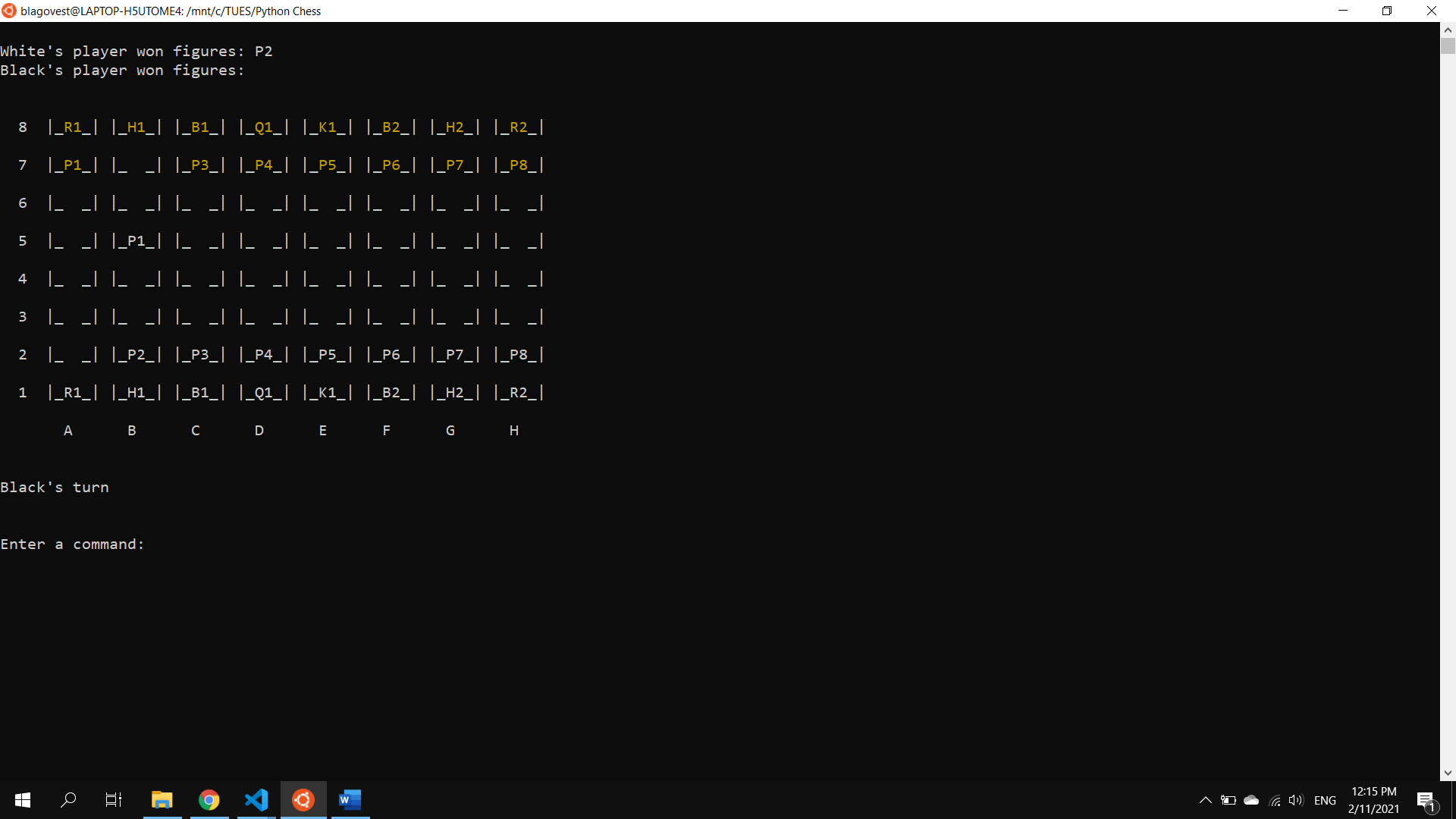
Играта на базово ниво се създава по следния начин:

* създава се обект *Game*, в който се инициализира всичко:
  + дъска от класа *ChessBoard*
  + 32 фигури от класа *Figure* (бели + черни)
  + двама играчи от класа *Player*, които взимат в себе си по 16 фигури от съответния за играча цвят

Сега ще разгледаме подробно как протича играта в метода *run* на *Game* класа.



**Фиг. 3.1.** Метод „run“ на обектa Game



**Фиг. 3.2.** Играта в Ubuntu терминал

На Фиг. 3.1. е описана в съкратен вариант последователността на действията, които се извършват в метода *run* на *Game* класа.

На Фиг. 3.2. е показана играта, стартирана в *Ubuntu* терминал.

Дъската се състои от две оси – вертикална (числова) и хоризонтална (азбучна). Използвайки тези две оси потребителят създава команди, например *„A2-A4“*, като този формат е единствения, който е поддържан (не се използват малки букви или друг тип символи).

Тестовете в класа *TestGame* са четири на брой:

* *test\_pawn( )* – тества движението и способността за взимане на фигури на пешка (тя е единствената фигура, която не се движи свободно във всички посоки, затова теста за нея е отделен)
* *test\_basic\_movement( )* – тества движението на всички типове фигури (без пешка)
* *test\_taking\_figures( )* – тества способността за взимане на фигури на всички типове фигури (без пешка)
* *test\_end( )* – тества установяването на край на играта

За да бъде възможно създаването на игра от външни ресурси (от променливи в тестовете) конструкторът на *Game* класа приема няколко аргумента:

* *w\_figs* – бели фигури
* *b\_figs* – черни фигури
* *board* – дъска

Ако тези аргументи са празни (празни масиви и *None* стойност) обекта сам ги инициализира.

За да бъде възможно тестването на ходове в играта метода *run* на *Game* класа приема един аргумент – *external commands*, който представлява масив от команди, които се изпълняват последователно, замествайки оригинално въвежданите на всеки ход команди от потребителя (в терминала). По този начин може да бъде изиграна цяла игра за част от секундата.

**3.2. Подробен преглед на базата данни**

****

**Табл. 3.1.** Потребители

Първата таблица (Табл. 3.1.) се казва *„users“* и съдържа основни данни за потребителите в платформата:

Лични атрибути:

* *id* – идентификационен номер (първичен ключ)
* *username* – име
* *password* – парола (хеширана)

Допълнителни атрибути (дали потребителят…):

* *is\_logged* – е влязъл в акаунта си
* *is\_waiting* – чака да започне игра
* *is\_playing* – играе в текущия момент

Атрибутите за чакане и за играене в текущия момент са взаимоизключващи се.

****

**Табл. 3.2.** Статистикa за потребителите

Втората таблица (Табл. 3.2.) е допълнение към първата и се казва *„user\_stats“*. В нея се запазват следните данни:

* *user\_id* – идентификационен номер на потребителя (външен ключ)
* *played\_games* – списък с идентификационните имена на всички игри, които потребителят е изиграл
* *win\_rate* – процент на победи

Поради липсата на възможност да се използват данни от тип масив в таблиците, в колоната за изиграните игри списъка се съхранява трансформиран под формата на символен низ, като при записване на нови данни в него той трябва да се върне в първоначалното си състояние на масив, за да бъдат възможни каквито и да е операции.

Процентът на спечелените игри се актуализира след всяка завършена игра на съответния потребител.

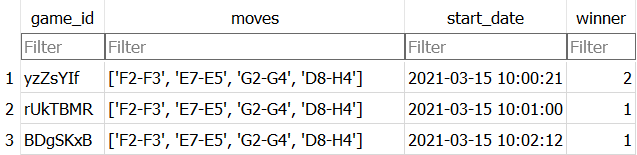
******

**Табл. 3.4.** Игри

Третата таблица (Табл. 3.3.) е за игрите (казва се *„games“*), като тя съдържа:

* *id* – идентификационно име (7 произволни символа – първичен ключ)
* *w\_player* – идентификационен номер на белия играч (външен ключ)
* *b\_player* – идентификационен номер на черния играч (външен ключ)
* *tournament\_id* – идентификационен номер на турнир, към който принадлежи играта (ако има такъв – външен ключ)

Идентификационния номер на турнир е записан под формата на *NULL* стойност ако играта не принадлежи към конкретен турнир.

****

**Табл. 3.4.** Детайли за игрите

Четвъртата таблица (Табл. 3.4.) е допълнение към *„games“* – *„game\_details“*. В нея се съдържат следните колони:

* *game\_id* – идентификационно име на играта (външен ключ)
* *moves* – списък със всички ходове, направени по време на играта
* *start\_date* – точната дата и час на започване на играта
* *winner* – идентификационен номер на победителя (ако има такъв - външен ключ)

Идентификационния номер на победителя в играта служи за ориентир когато искаме да разберем дали тя е приключила – ако е все още активна, стойността е от тип *NULL*.

****

**Табл. 3.5.** Турнири

Следващата таблица (Табл. 3.5.) се казва *„tournaments“* и съдържа данни за създадените турнири.

Колоните са следните:

* *id* – идентификационен номер на турнира
* *waiting\_users* – списък с идентификационните номера на чакащите потребители за започване на турнира (щом започне списъка бива изтрит)
* *quarter\_final* – списък с идентификационните номера на потребителите, участващи в четвърт финал
* *semi\_final* – списък с идентификационните номера на потребителите, участващи в полуфинал
* *final* – списък с идентификационните номера на потребителите, участващи във финал
* *winner* – идентификационен номер на потребителя, който е спечелил турнира

****

**Табл. 3.6.** Съобщения

Последната таблица (Табл. 3.6.) се казва *„messages“* и съдържа данни за всички съобщения, изпратени в чат по време на игрите. Тя притежава следните атрибути:

* *id –* идентификационен номер на съобщението (първичен ключ)
* *user\_id –* идентификационен номер на потребителя, който е изпратил съобщението (външен ключ)
* *game\_id –* идентификационно име на играта, в която потребителят е изпратил съобщението
* *text –* текстът на самото съобщение
* *time –* датата и часа, в които съобщението е било изпратено

**3.3. Същност и детайли на WEB приложението за игра на шах**

Сървърната част на приложението, написана на *Python*, с помощта на *Flask* и *SQLAlchemy*, е разпределена в осем файла (с разширение .py):

* *init* – в него се инициализира *Flask* приложението; от него всички други файлове взимат самото приложение чрез *import*
* *database* – инициализира базата данни
* *auth* – обслужва функционалностите, свързани с управление на профилите на потребителитe
* *main* – обслужва общи функционалности, както и профилната страница
* *game\_base* – отнася се конкретно към потребителската игра
* *multiplayer* – обслужва функционалностите, свързани с играта от тип обикновен мултиплеър
* *tournament* – насочен е към функционалностите, свързани с турнир
* *models* – съдържа всички *SQLAlchemy* структури, необходими за създаване на базата данни

Сега ще разгледаме всеки един от тези файлове по-подробно (по-важните им части).

**3.3.1.***init.py –* съдържа в себе си три метода:

- *create\_app( )* - създава приложението, като конфигурира път за база данни, сесия и логин мениджър; извиква се веднъж в *main.py*

- *get\_app( )* – връща като стойност самото приложение; използва се във всички останали файлове, където то е нужно

- *get\_random\_string( )* – създава променлива от тип символен низ със седем произволни букви (малки и големи), която се използва като идентификационно име за игра във функционалностите на другите файлове

**3.3.2.** *database.py* – създава базата данни и сесия към нея и съдържа един метод – *init\_db( )*, който създава мета-данни за базата; от този файл се взима променливата за сесия на базата данни за всички останали функционалности

**3.3.3.** *auth.py* – съдържа следните функционалности:

- регистрация

- влизане и излизане от профил

- проверка дали потребителя е влязъл в профил

**3.3.4.** *models.py* – тук се намират структурите на *SQLAlchemy* под формата на класове (създаващи таблиците в базата данни):

- *User* – създава потребител

- *userStats* – допълва потребителския клас

- *GameT* – създава игра (не се казва *Game*, защото вече съществува клас с това име, който създава *Python* играта)

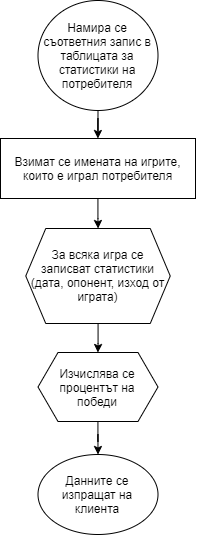
- *gameDetails* – допълва класа за играта

- *Tournament* – създава турнир

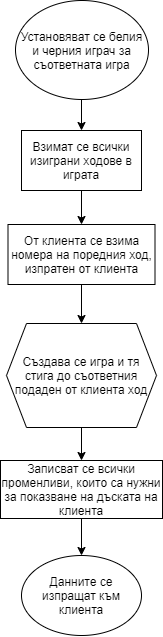
- *Message* – създава съобщение

**В следващите ключови за проекта файлове, за всички важни функционалности са направени блок-схеми, които показват логиката и последователността им на действие.**

**3.3.5.** *main.py* – разгледани функционалности - профилна страница (Фиг. 3.3.) и за страница за преглеждане на приключила игра по ходове (Фиг. 3.4.).



**Фиг. 3.3.** Профилна страница

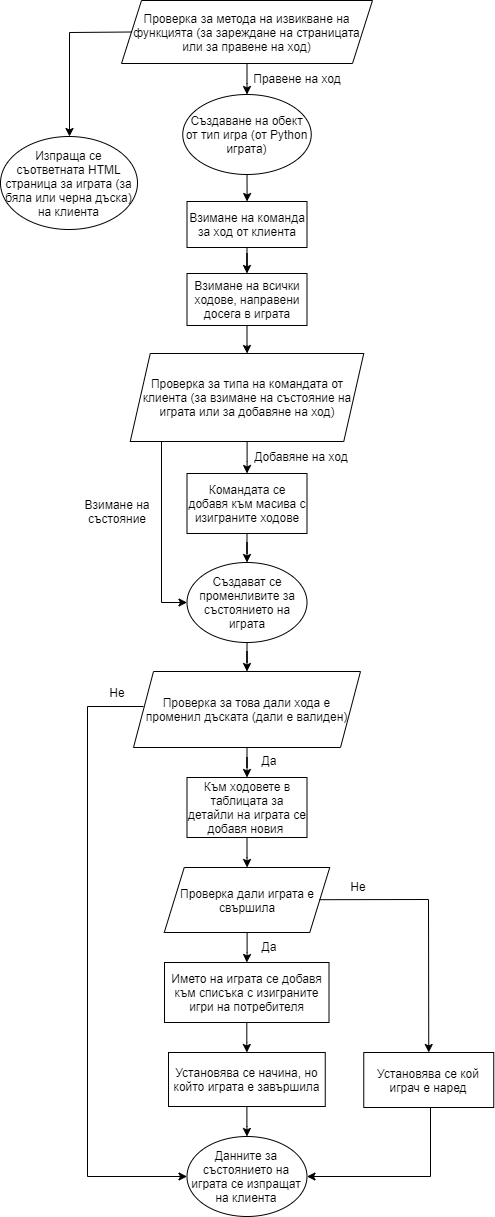


**Фиг. 3.4.** Страница за преглеждане на изиграни игри

**3.3.6.** *game\_base.py* – разгледани функционалности – чат по време на игра (Фиг. 3.5.) и самото протичане на игра (Фиг. 3.6.).

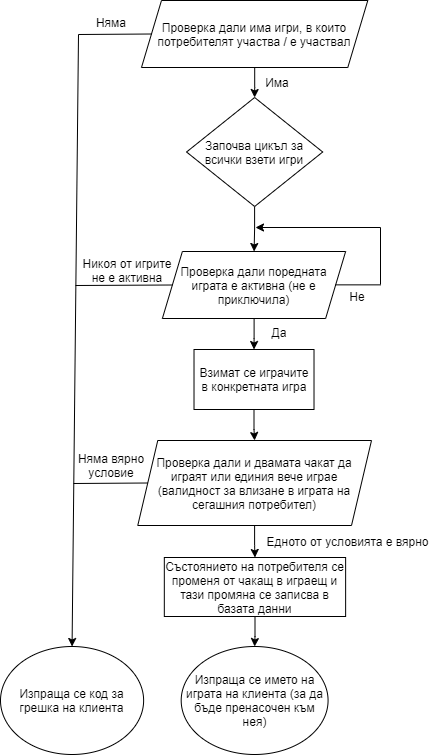
****

**Фиг. 3.5.** Чат по време на игра

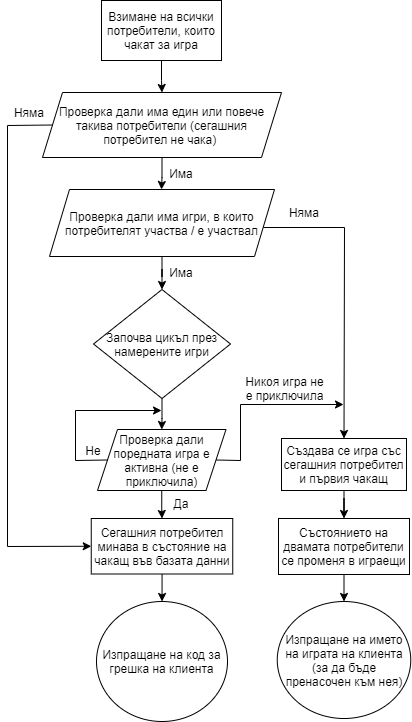
****

**Фиг. 3.6.** Протичане на игра

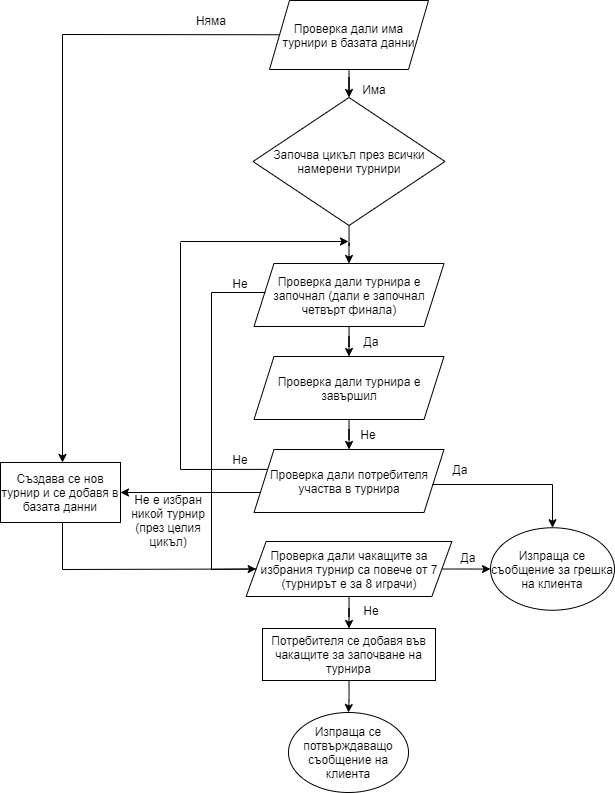
**3.3.7.** *multiplayer.py* – разгледани функционалности – влизане в игра (Фиг. 3.7.) и създаване на игра (Фиг. 3.8.).

****

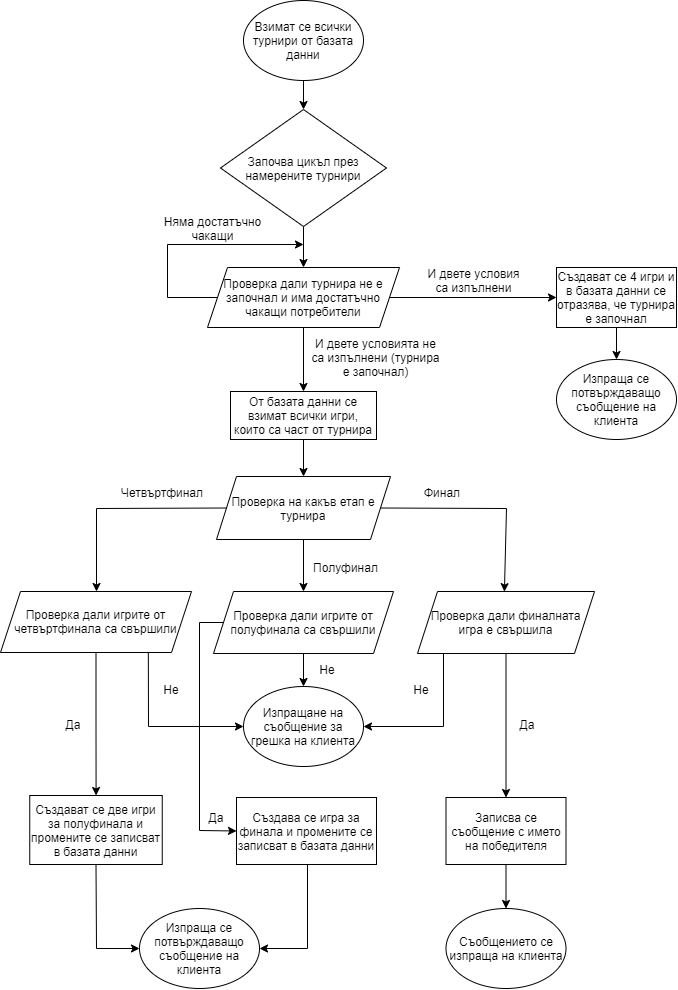
**Фиг. 3.7.** Влизане в игра

****

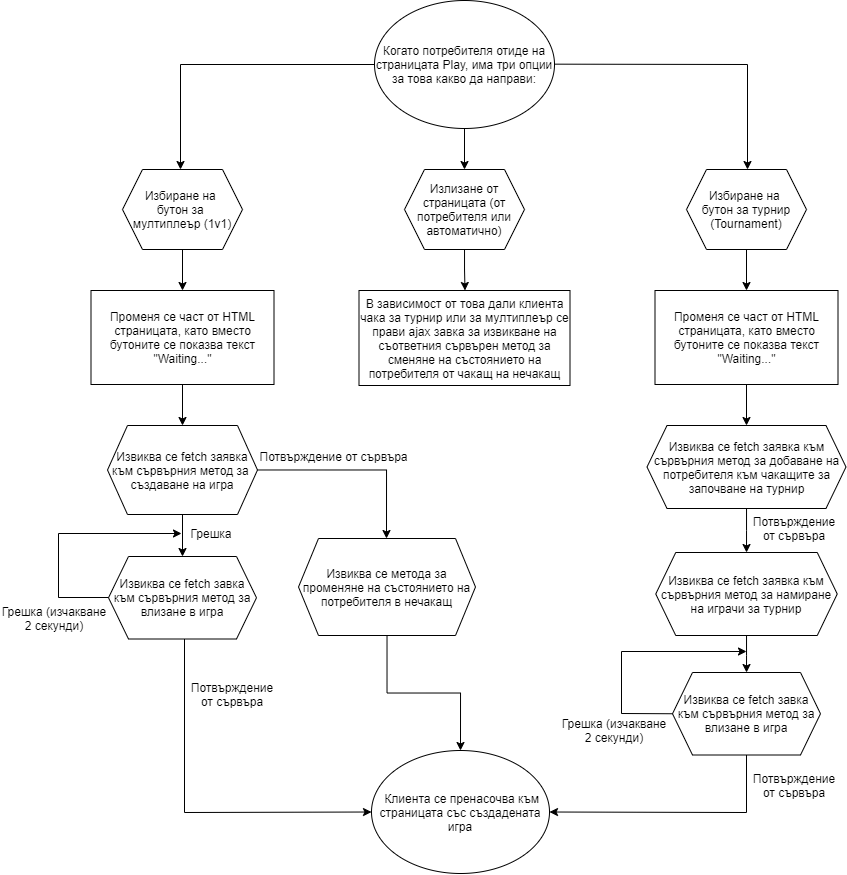
**Фиг. 3.8.** Влизане в игра

**3.3.8.** *tournament.py* – разгледани функционалности – намирането на играчи за турнир (Фиг. 3.9.) и изчисляването на следващи кръгове на турнира (Фиг. 3.10.).

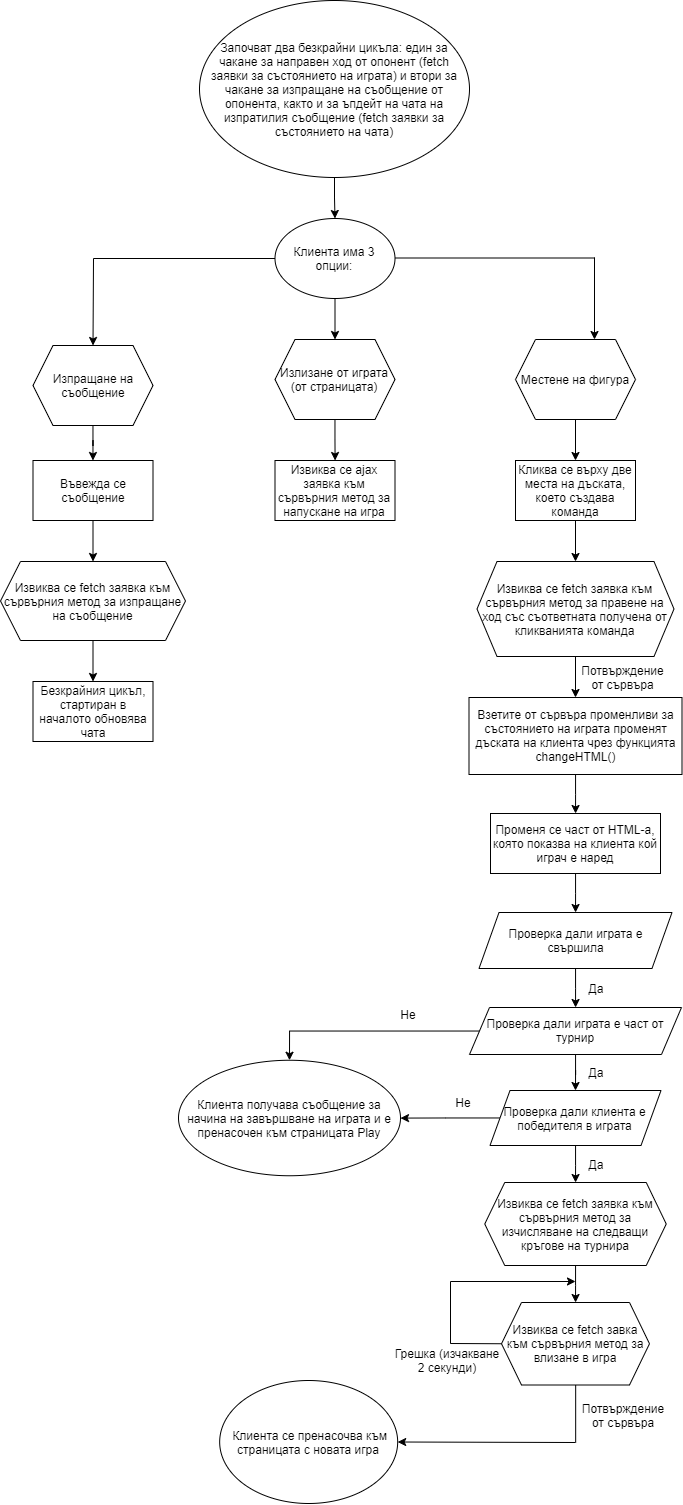
**Фиг. 3.9.** Намиране на играчи за турнир

****

**Фиг. 3.10.** Изчисляване на следващия кръг в турнира

**3.3.9.** Последните файлове, които съдържат важни функционалности са *play.js* и *game.js*. Те използват сървърните функционалности и добавят свои за оптимален потребителски интерфейс. Структурата и работата на двата файла са представени като блок-схеми (Фиг. 3.11. и 3.12.).

**Фиг. 3.11.** Функционалност на файла “play.js”

****

**Фиг. 3.12.** Функционалност на файла “game.js”

**3.4. Извадки от код с особено значение**

**3.4.1. Проверка за предотвратяване на състояние шахмат**

****

**Фиг. 3.13.** Проверка за предотвратяване на състояние шахмат

Обяснение на логиката и последователността от действия:

Има играч, който е в шахмат, и той трябва да провери дали някоя фигура не може да го защити. Минава се през всички негови фигури и се пробват всички възможни следващи ходове за всяка от фигурите, тоест всяко място, на което дадената фигура може да бъде преместена на следващия ход. Тестването на това местене на фигури не трябва да променя сегашното състояние на дъската, затова се създава променлива – копие на дъската и върху нея се тества. След всяко преместване се проверява дали царят е в шах при промененото състояние на дъската. Щом в един от проверените случаи царят не е в шах, тогава предполагаемото условие за мат отпада.

**Фиг. 3.14.** Проверка за установяване на състояние мат

**3.4.2. Проверка за установяване на състояние мат******

Обяснение на логиката и последователността от действия:

Минава се през фигурите на играча, който в момента е на ред, и се взима царя. Както всяка фигура, царят има атрибут за възможни свои бъдещи позиции по дъската, валидни за един следващ ход; този атрибут се актуализира. Минава се през всички възможни следващи позиции на царя, и при всяка от тях се минава и през фигурите на опонента, или по-точно през възможните позиции на всяка от тях.

Взимайки възможна позиция от фигура на опонента и такава от тези на царя, при съвпадение броячът за мат се увеличава. Състоянието за мат се зачита за вярно, ако този брояч е равен на броя на възможните позиции на царя, тоест всяка от тези позиции вкарва царя в шах.

Тук съществува един проблем – пешката не смята за възможни позиции диагоналните, освен ако няма фигура върху тях. Това означава, че ако някоя пешка поставя царя в шах в някоя от неговите възможни позиции, алгоритъмът няма да го засече, понеже се сравняват единствено цифри и букви за позициите, а не самата дъска.

За да бъде разрешен този проблем, когато се минава през фигурите на опонента и наред дойде пешка, се прави следното: създава се копие на сегашната дъска, за да се избегнат нежелани промени по основната, взима се царя, който е във фигурите на текущия играч и се премества на съответната му възможна позиция, която е взета в момента; после се актуализират възможните позиции на пешката, за да се запазят промените, които са направени. Така възможните позиции на пешката вече могат да бъдат сравнени с тези на царя и да се разбере дали царя попада в състоянието мат.

**ЧЕТВЪРТА ГЛАВА**

**РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ**

Това ръководство е ориентирано главно към потребителите на някаква форма на операционна система *Linux* (*Ubuntu*-базирана)

Отворете програмата терминал. За да инсталирате нужните програми, пакети и модули използвайте командите по-долу:

*Python*:

$ sudo apt update

$ sudo apt install software-properties-common

$ sudo add-apt-repository ppa:deadsnakes/ppa

$ sudo apt install python3.9

*Pip*:

$ pip install pip

*Flask*:

$ pip install Flask

$ pip install Flask-Login

$ pip install Flask-Session

$ pip install Flask-API

*SQLAlchemy*:

$ pip install SQLAlchemy

$ pip install Flask-SQLAlchemy

*Werkzeug*:

$ pip install Werkzeug

Трябва да създадете две *„env“* променливи във файла *“.bashrc”*.

Първо отваряте файла през терминала, например с командата:

$ nano ~/.bashrc

След това отивате на най-долния ред на файла и създавате две променливи – *„CHESS\_DATABASE“* и *„PYTHON\_GAME“*, като те ще съдържат пътищата до съответните файлови ресурси. Последната папка *“TUES”* в примерните пътища е папката на *Github* репозиторито. Пътищата могат да варират в зависимост от системата Ви, така че показания по-долу пример е само образец.

export CHESS\_DATABASE='sqlite:///C:\\TUES\\Github\\TUES\\chess.db'

export PYTHON\_GAME='/mnt/c/TUES/Github/TUES/python\_game'

След това запишете файла с командите *Ctrl+X*, *Y*, *Enter*.

За да се сигурни, че променливите са запазени трайно в системата, изпълнете следната команда в терминала:

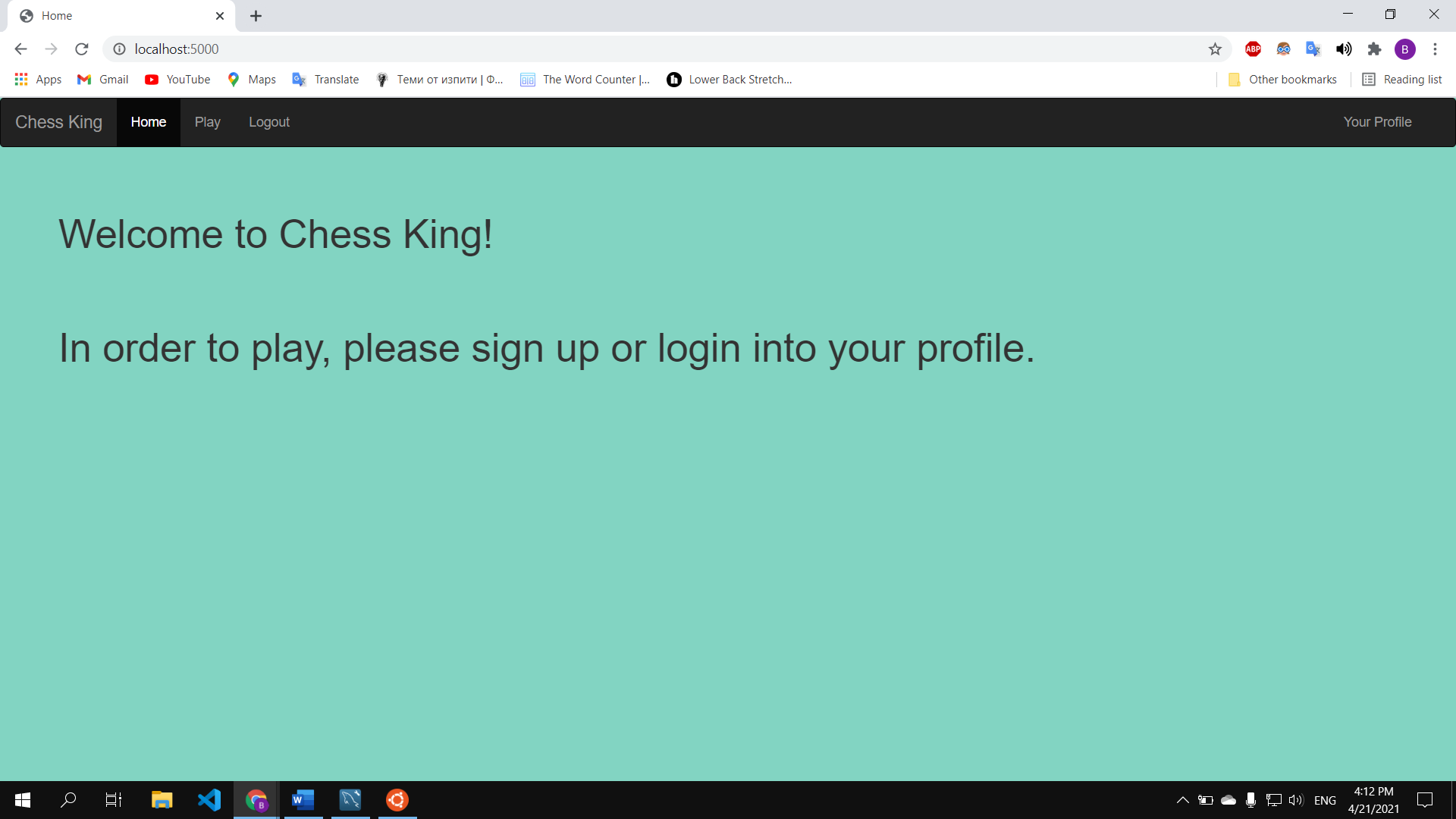
$ source ~/.bashrc

Вече сте готови да стартирате приложението. Влезте в папката, където са изходните ресурси и изпълнете командата:

$ FLASK\_APP=main flask run

Ако искате да терминирате приложението, използвайте *Ctrl+C*.

**Фиг. 4.1.** Начална страница

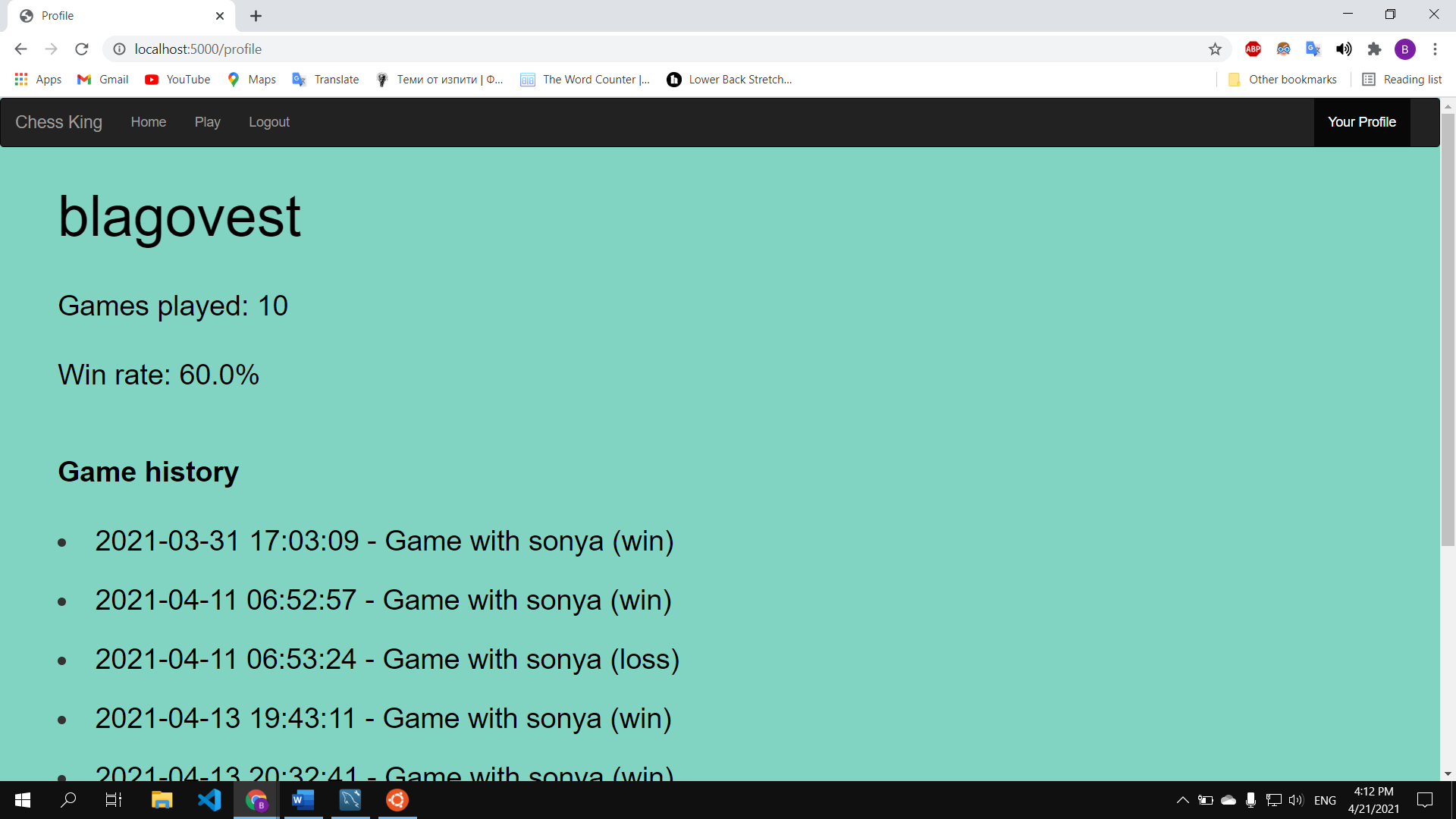


Щом пуснете приложението, заредете адреса на *“localhost:5000”* в браузъра си (в случая порта е 5000, но това може да варира) и ще се озовете на началната страница (Фиг. 4.1.).

За да започнете игра първо трябва да влезете в профила си. Кликнете на бутона *„Login“* от менюто. За да създадете профил, последвайте линка *„Sign up“* под полетата.

Щом се регистрирате, автоматично ще бъдете пренасочени към *“Login”* страницата. След въвеждане на данните си, ще бъдете пренасочени към началната страница. Ако това не се случи, значи въведените от Вас данни не са правилни (ще Ви излезе съобщение за грешка).

Докато не сте влезли в профила си, началната страница *„Home“*, страницата за регистрация и *„Login“* страницата са „отключени“, докато всички други страници отказват достъп и автоматично Ви препращат към *„Login“*. По този начин се контролира свободата и коректността на потребителите в сайта.



**Фиг. 4.2.** Профилна страница

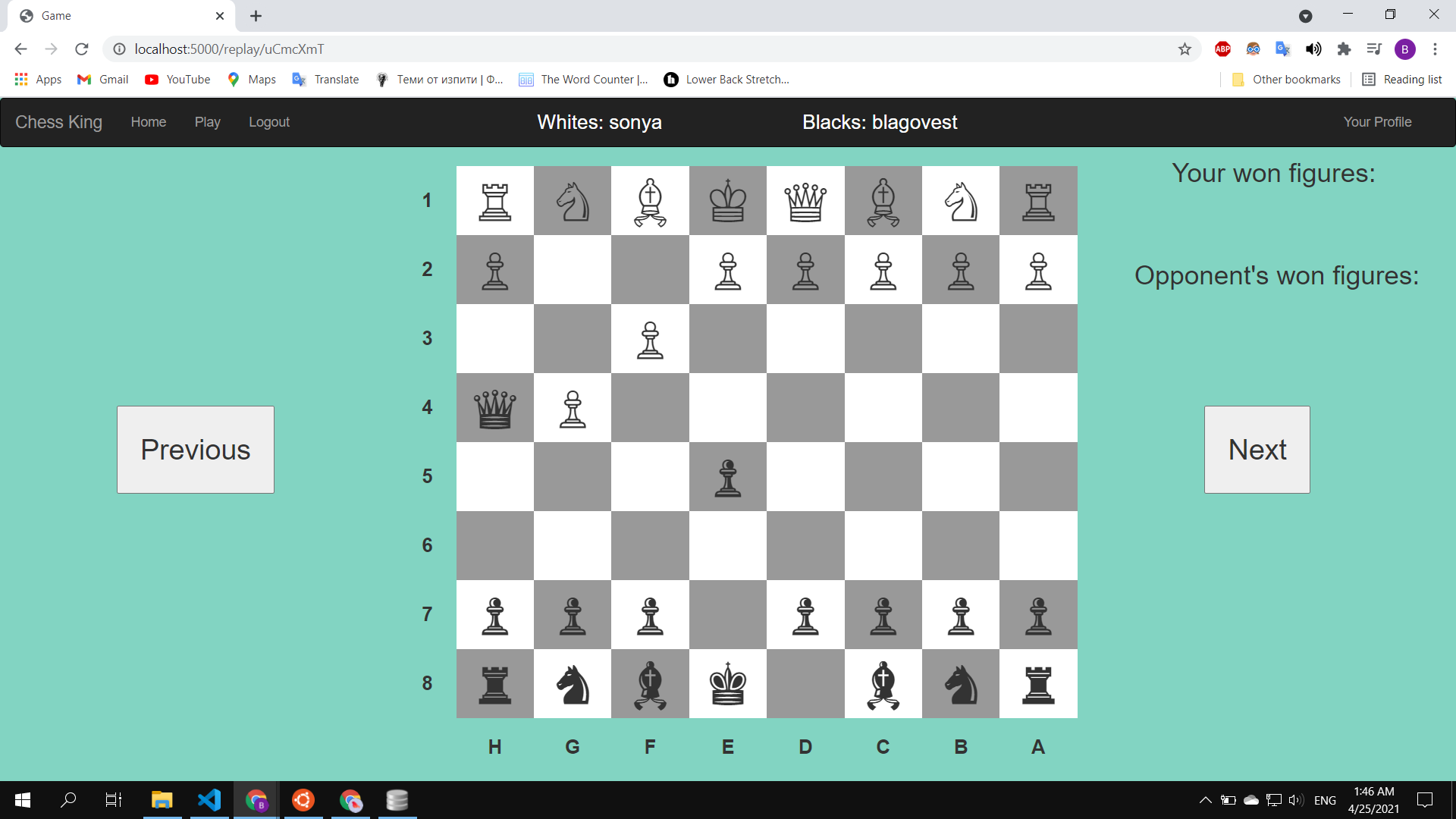
Профилната страница (Фиг. 4.2.) се състои от:

* име на потребителя
* брой на изиграни игри
* процент на спечелени игри
* история на игри

Историята на игри се състои от списък на изиграните досега игри със следните характеристики:

* датата и часът на започване на играта
* името на потребителя, който е бил опонент
* дали играта е била спечелена, загубена или на равно.

Всяка игра от списъка представлява линк към страница, където може да бъде проследена ход по ход.

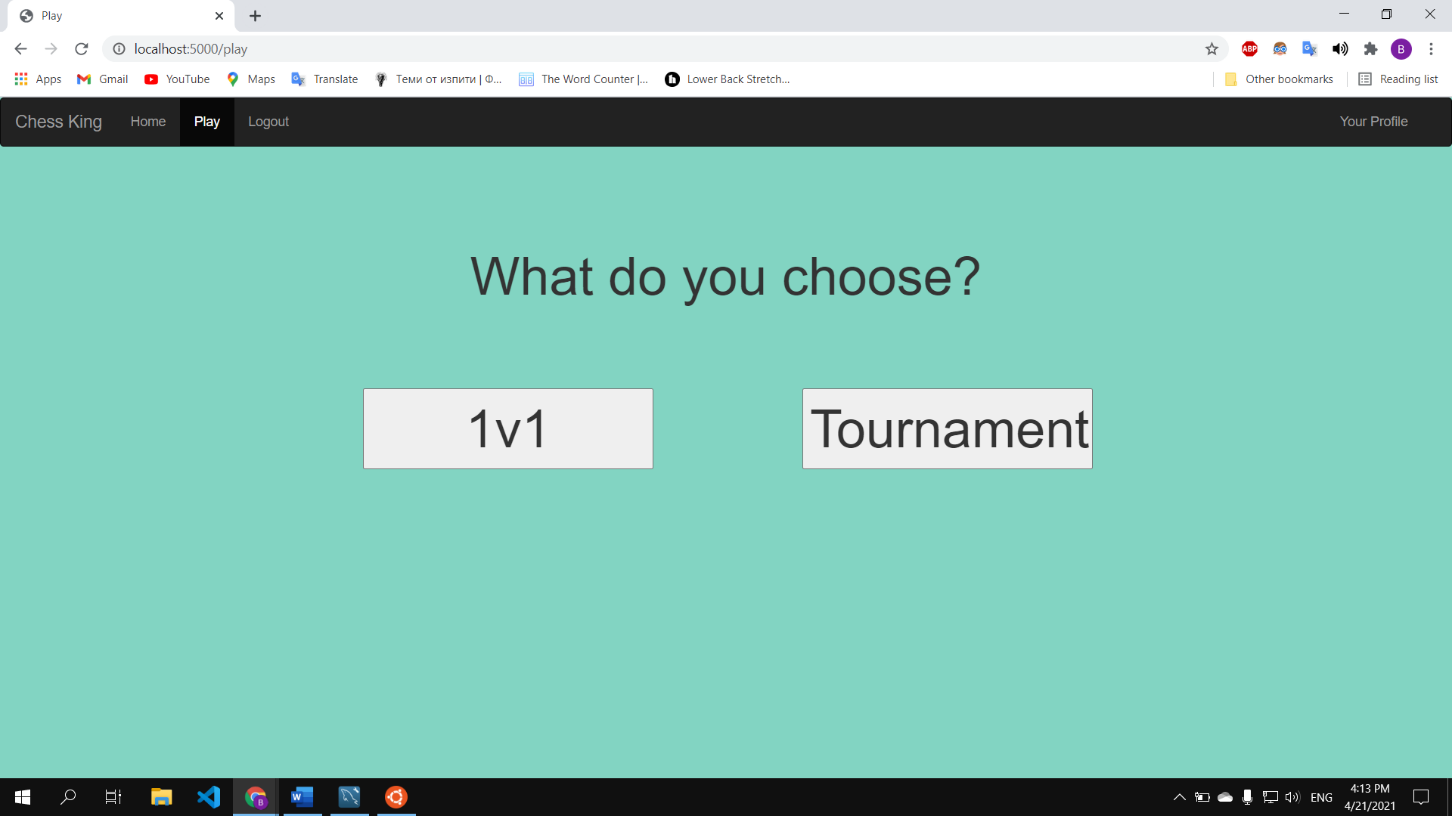


**Фиг. 4.3.** Страница за разглеждане на изиграна игра

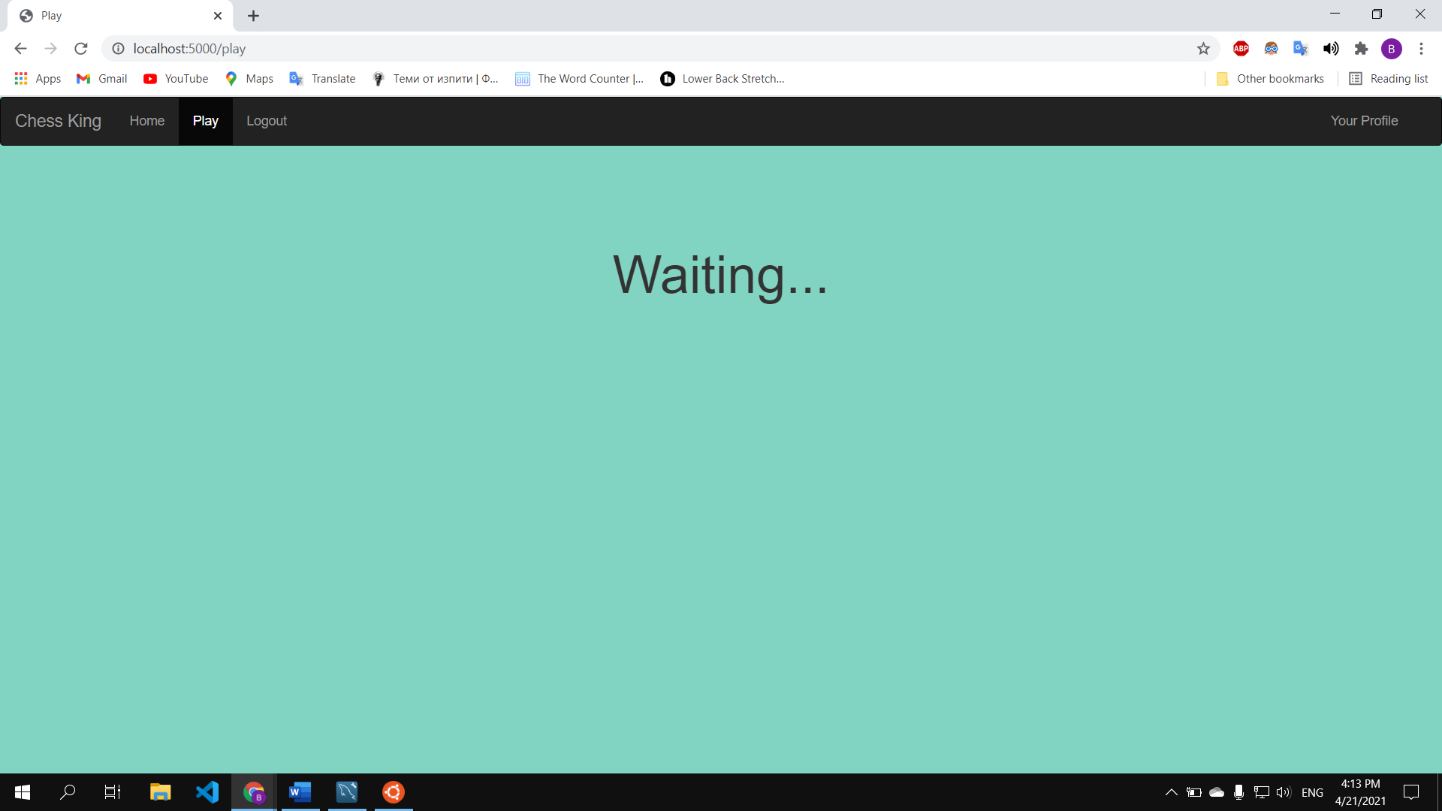
Над дъската (Фиг. 4.3.) пише кой играч с кои фигури е, а бутоните от двете страни преместват фигурите по ходовете, които са изиграни по време на играта. Над бутона за следващ ход се показват взетите досега фигури и от двамата играчи.

Когато искате да започнете игра, трябва да натиснете бутона *„Play“* от горното меню.

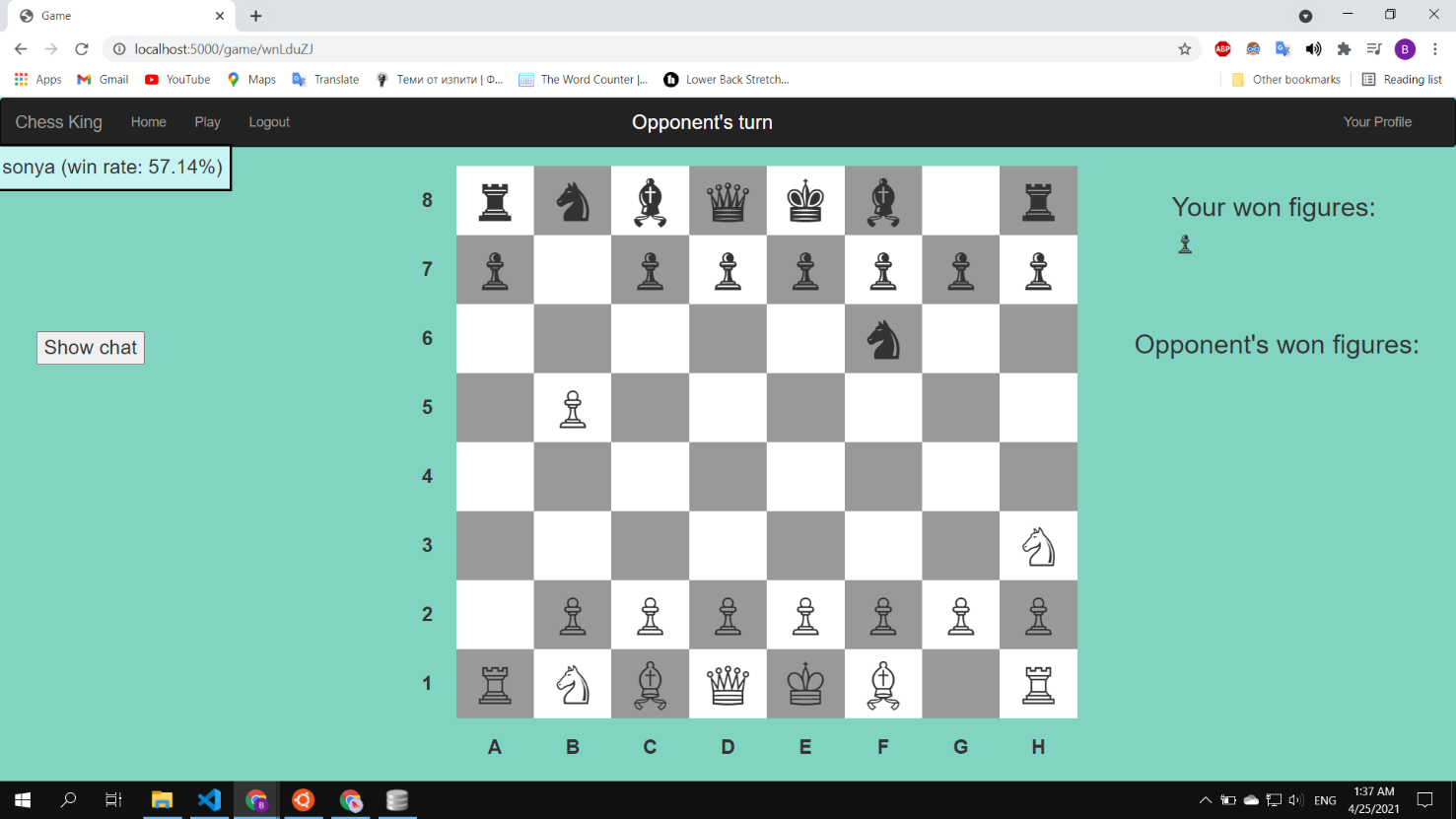
На страницата, която се зарежда (Фиг. 4.4.) имате две опции – да играете в нормален мултиплеър или в турнир.

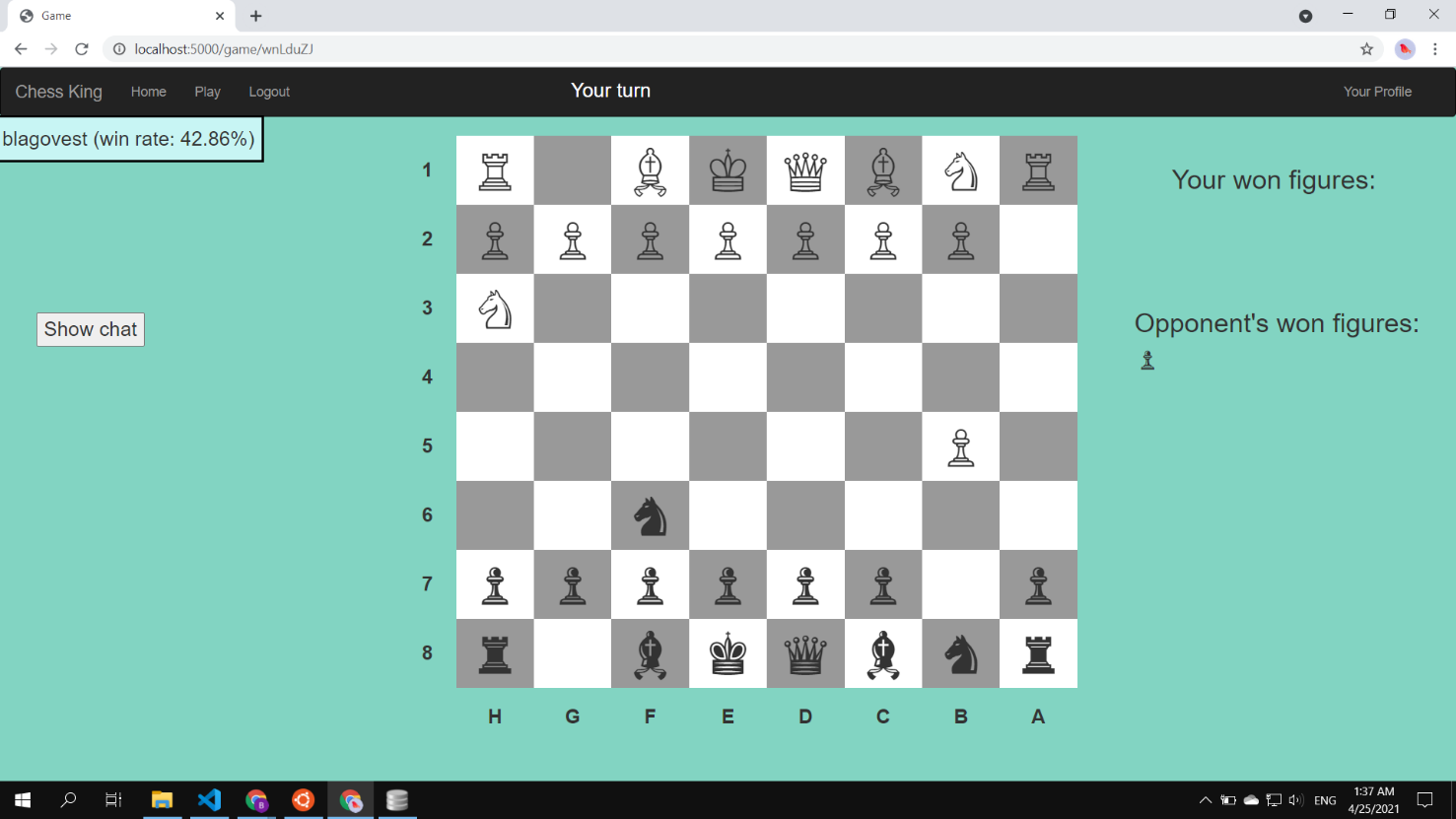
****

**Фиг. 4.4.** Страница за избиране на режим на игра

Избирайки една от опциите, влизате в списъка с чакащи играчи за съответния тип игра.

**Фиг. 4.5.** Променена страница за избиране на режим за игра (режим чакане)

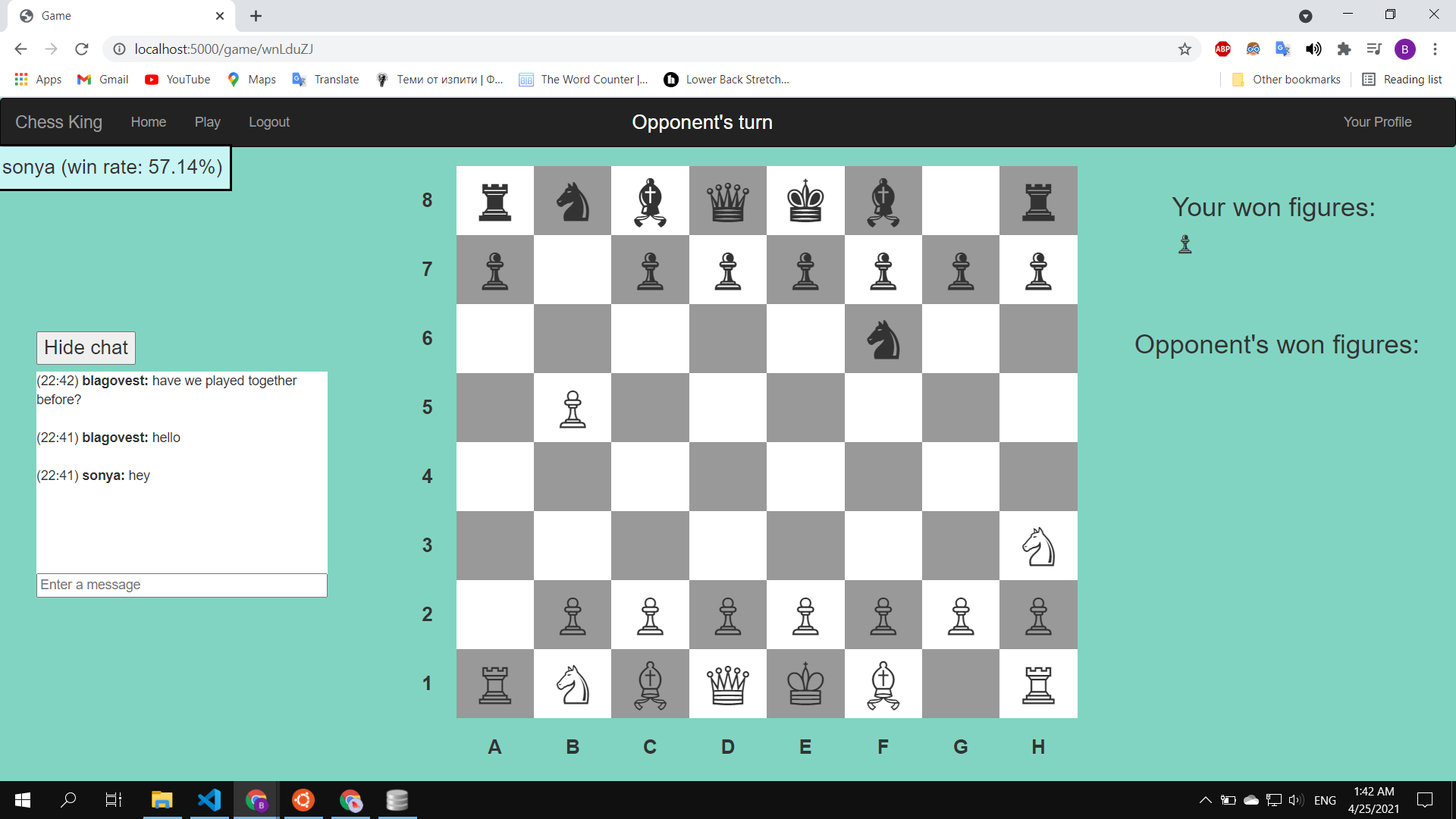
Нормалният мултиплеър започва при двама чакащи, а турнирът – при осем.



**Фиг. 4.5.** Страница по време на игра на белия играч

**Фиг. 4.6.** Страница по време на игра на черния играч

На Фиг. 4.6. и 4.7. са показани екраните на играча с белите фигури и на този с черните.

На Фиг. 4.8. бутонът вляво служи за показване и скриване на чата по време на играта.

**Фиг. 4.7.** Страница по време на игра със показан чат

Съобщенията се показват чрез следните параметри:

* час и минута на изпращане
* име на потребителя, който е изпратил съобщението
* текст на съобщението

Съобщенията са сортирани по време на изпращане в низходящ ред, тоест най-скорошните се появяват първи.

**ЗАКЛЮЧЕНИЕ**

**Обобщение**

Чрез разработването на този проект бе постигната една напълно функционална реализация на WEB приложение за игра на шах, като самата игра представлява ядрото, а сървърът и клиентската част са нужни, за да може играта да бъде използвана по лесен, достъпен и приятен за потребителите начин.

**Бъдещи подобрения**

За да бъде възможна реализацията на проекта на пазарно ниво трябва да се извършат/добавят следните промени/допълнителни имплементации:

* да се напишат тестове за сървърните и клиентските функционалности
* да се подобри ефективността на софтуера и максималния размер на данни, които могат да бъдат контролирани адекватно от системата
* да се имплементира някакъв вид комуникация между потребителите извън конкретна игра, например под формата на отбори от играчи и възможност за покана за игра между отделните потребители
* допълнителна имплементация би могла да бъде изкуствен интелект под формата на бот, който да играе с потребителя на избрано ниво на трудност
* потребителите да се класифицират в лиги, което да улесни намирането на най-добрите играчи в платформата

**ИЗПОЛЗВАНА ЛИТЕРАТУРА**

1. PHP - <https://www.php.net/docs.php>
2. Python - <https://docs.python.org/3/>
3. Java - <https://docs.oracle.com/en/java/>
4. MySQL - <https://dev.mysql.com/doc/>
5. SQL Server - <https://docs.microsoft.com/en-us/sql/sql-server/>
6. PostgreSQL - <https://www.postgresql.org/docs/>
7. HTML - <https://devdocs.io/html/>
8. CSS - <https://devdocs.io/css/>
9. JavaScript - <https://devdocs.io/javascript/>
10. Laravel - <https://laravel.com/docs/8.x>
11. CodeIgniter - <https://codeigniter.com/docs>
12. Django - <https://docs.djangoproject.com/en/3.1/>
13. Flask - <https://flask.palletsprojects.com/en/1.1.x/>
14. Spring - [https://docs.spring.io/spring-framework/docs/5.3.5-SNAPSHOT/reference](https://docs.spring.io/spring-framework/docs/5.3.5-SNAPSHOT/reference%20)
15. Struts - <https://struts.apache.org/docs/>
16. Angular - <https://angular.io/docs>
17. React - <https://reactjs.org/docs/getting-started.html>
18. Vue - <https://vuejs.org/v2/guide/>
19. jQuery - <https://api.jquery.com/>
20. “chess.com” - <https://www.chess.com/>
21. “lichess.org” - <https://lichess.org/>
22. SQLite - <https://sqlite.org/docs.html>
23. DB Browser for SQLite - <https://github.com/sqlitebrowser/sqlitebrowser/wiki>
24. SQLAlchemy - <https://docs.sqlalchemy.org/en/13/>
25. Bootstrap - <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
26. Visual Studio Code - <https://code.visualstudio.com/docs>
27. Ubuntu WSL 2 - <https://docs.microsoft.com/en-us/windows/wsl/>
28. Git - <https://git-scm.com/doc>
29. Github - <https://docs.github.com/en>

**СЪДЪРЖАНИЕ**

**Увод**………………………………………………………………………………. **3**

**1. Първа глава** – Методи и технологии за реализиране на WEB приложения

1.1. Основни принципи, технологии и развойни среди за реализиране на WEB приложения…………………………………………………………………......... **4**

1.2. Съществуващи решения и реализации…………………………………….. **6**

**2. Втора глава** – Проектиране на структурата на WEB приложение за игра на шах

2.1. Функционални изисквания към WEB приложение за игра на шах……… **9**

2.2. Съображения за избор на програмни средства и развойната среда…….. **10**

2.3. Файлова структура на проекта……………………………………………. **12**

2.4. Основна структура и принцип на действие на WEB приложение за игра на шах…………………………………………………………………………… **13**

2.5. Структура на базата от данни…………………………………………….. **14**

**3. Трета глава** – Програмна реализация на WEB приложение за игра на шах

3.1. Ядрото на проекта – играта на шах………………………………………. **15**

3.2. Подробен преглед на базата данни……………………………………….. **19**

3.3. Същност и детайли на WEB приложението……………………………… **25**

3.4. Извадки от код с особено значение………………………………………. **37**

**4. Четвърта глава** – Ръководство на потребителя…………………………... **40**

**5. Заключение**………………………………………………………………….. **49**

**6. Използвана литература**……………………………………………………. **50**