**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

**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 за имплементирането на 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 десетки разработени и публични имплементации, което изглежда сякаш намалява значителността на идеята. Решението обаче е да се направи 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 прави проекта интересен и също така 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.

Сървъра 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 и 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 в таблици, а чрез приложението тази информация бива създадена, редактирана и извиквана от клиента за по-нататъшна обработка.

Страната на клиента представлява WEB частта на приложението, на практика уебсайта. Той е изграден чрез сбор 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]

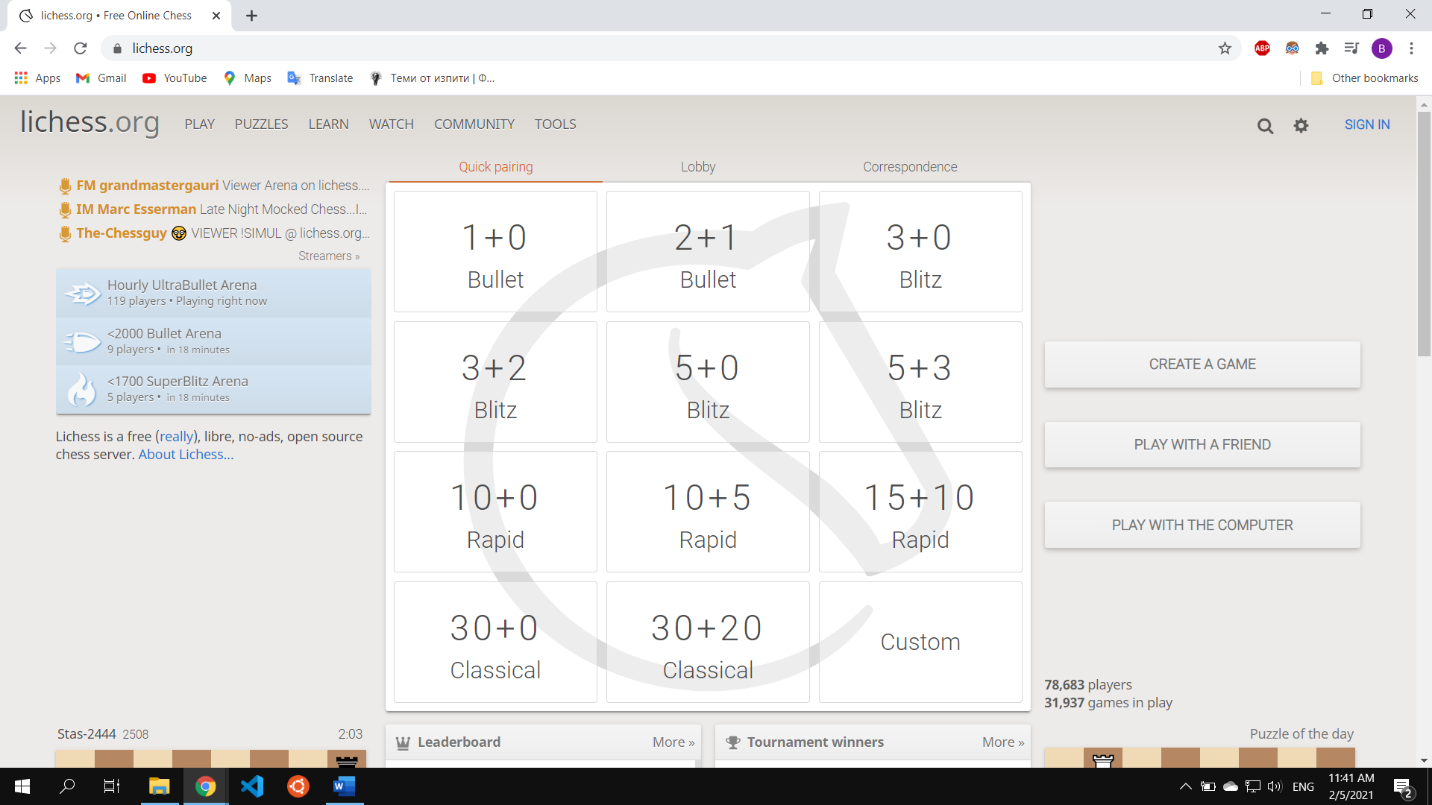
За клиентската част (JavaScript):

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

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

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

****

****

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

**Табл. 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 ще разгледаме двете най-използвани – “chess.com” [19] (Фиг. 1.1.) и “lichess.org” [20] (Фиг. 1.2.). Двете платформи имат много сходни характеристики и функционалности, но все пак имат 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 почти постоянно 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 и обратното - зареждането им в сайта
* турнири
* различни настройки за времето, за което се изпълнява ход в игра
* 8 различни вида игра на шах
* игра срещу компютър
* компютърен треньор, работещ под формата на пъзели за решаване и рейтинг на развитието на потребителя
* голям набор от обучителни видеа

В общите функционалности “chess.com” превъзхожда с малка разлика “lichess.org” във частта със 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. Платформата съперник 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

- Играене на шах в мултиплеър

- История на изиграните игри

- Рейтинг система на играчите

- Поддръжка на турнири

- Чат между играчите по време на играта

Обобщение: приложението трябва да представлява сайт, в който да се регистрират и да влизат в профила си потребители, които да могат да играят на 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 формата на турнир. Трябва да има рейтинг на 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 избрах за дипломната си работа са следните: Python, Flask, SQLite [21], SQLAlchemy [22], HTML, CSS, JavaScript, Bootstrap [23].

Python е един от най-популярните, развити и функционални програмни езици. Този език е избран поради красивия му интерфейс, голямата 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 за 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. 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 останалите части на проекта и базата данни.

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 добре вписващ се с останалите елементи дизайн, който реално е използвания в проекта. 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 [24] и Ubuntu WSL 2 [25].

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

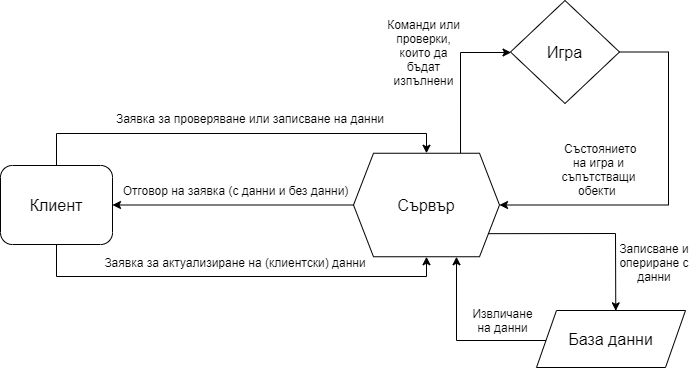
Ubuntu WSL 2, който се използва на Windows 10 представлява алтернатива на вградения в операционната система Linux Ubuntu терминал, като съдържа в себе си същите функционалности и улеснява работата с Flask и Git [26], както и цялостното управление на компютърната система. Поради сложността на инсталиране и конфигуриране на Linux операционна система, нуждата от терминал не оправдава такова начинание, затова е избран по-лесния вариант чрез интеграция с WSL 2, която се инсталира лесно и работи достатъчно добре като заместител, като на повърхността не личат никакви отрицателни страни на това решение.

**2.3. Цялостна структура на файловете в приложението**

**├── 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

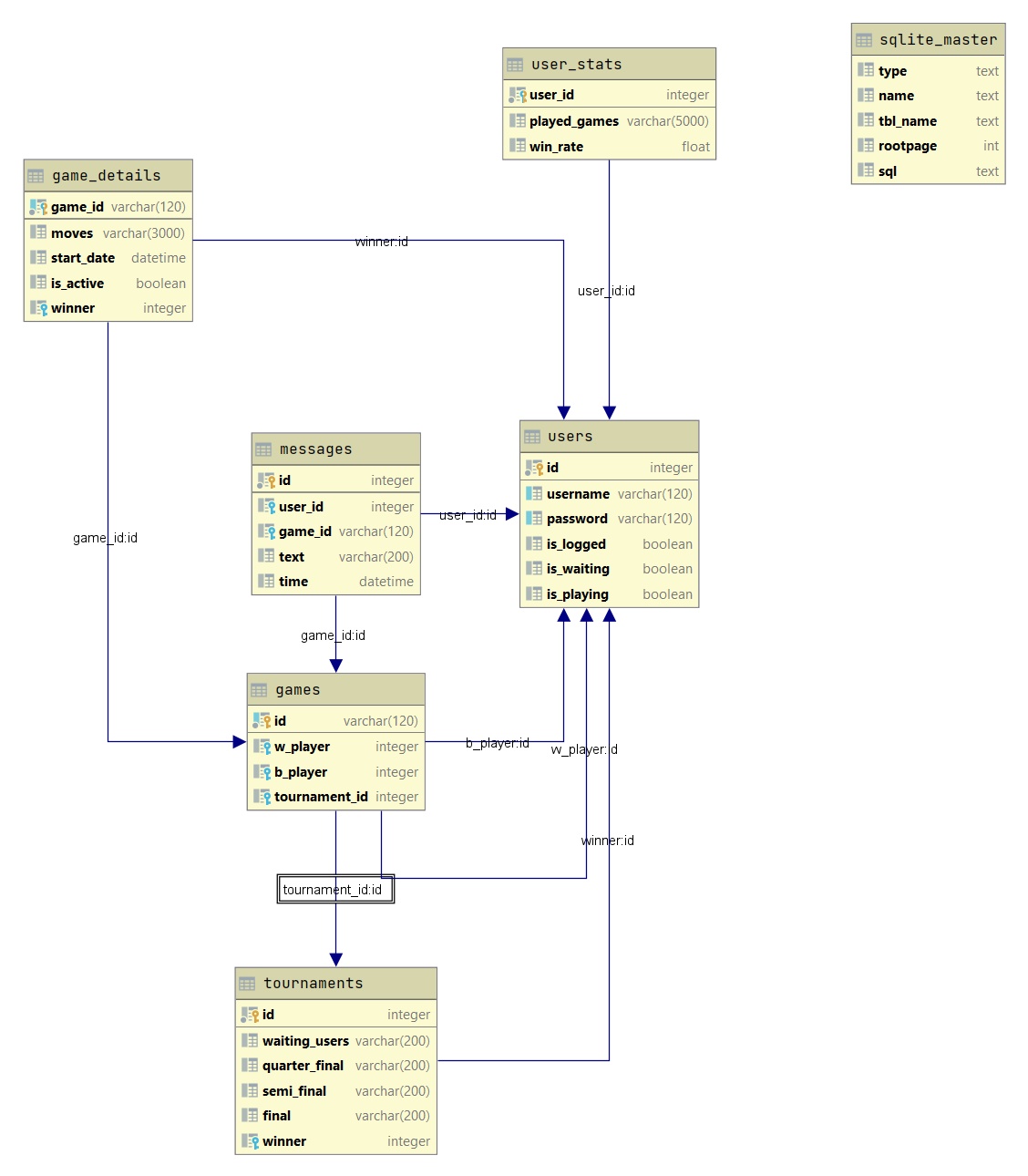
В папката „python\_game“ (Фиг. 2.1.) се намират файловете, представляващи самата игра на шах, в папката „static“ се намират CSS, JavaScript и Bootstrap файловете, разделени в папки, а в „templates“ са HTML файловете.

**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 не 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. Проектиране на структурата 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 командна Python програма, 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* – Фиг. на даден играч
* *Player* – играч
* *Game* – събирателен клас, който изпълнява цялостния процес, който наричаме „игра“
* *TestGame* – unittest-ове, които проверяват коректната работа на играта

Играта на базово ниво се създава по следния начин:

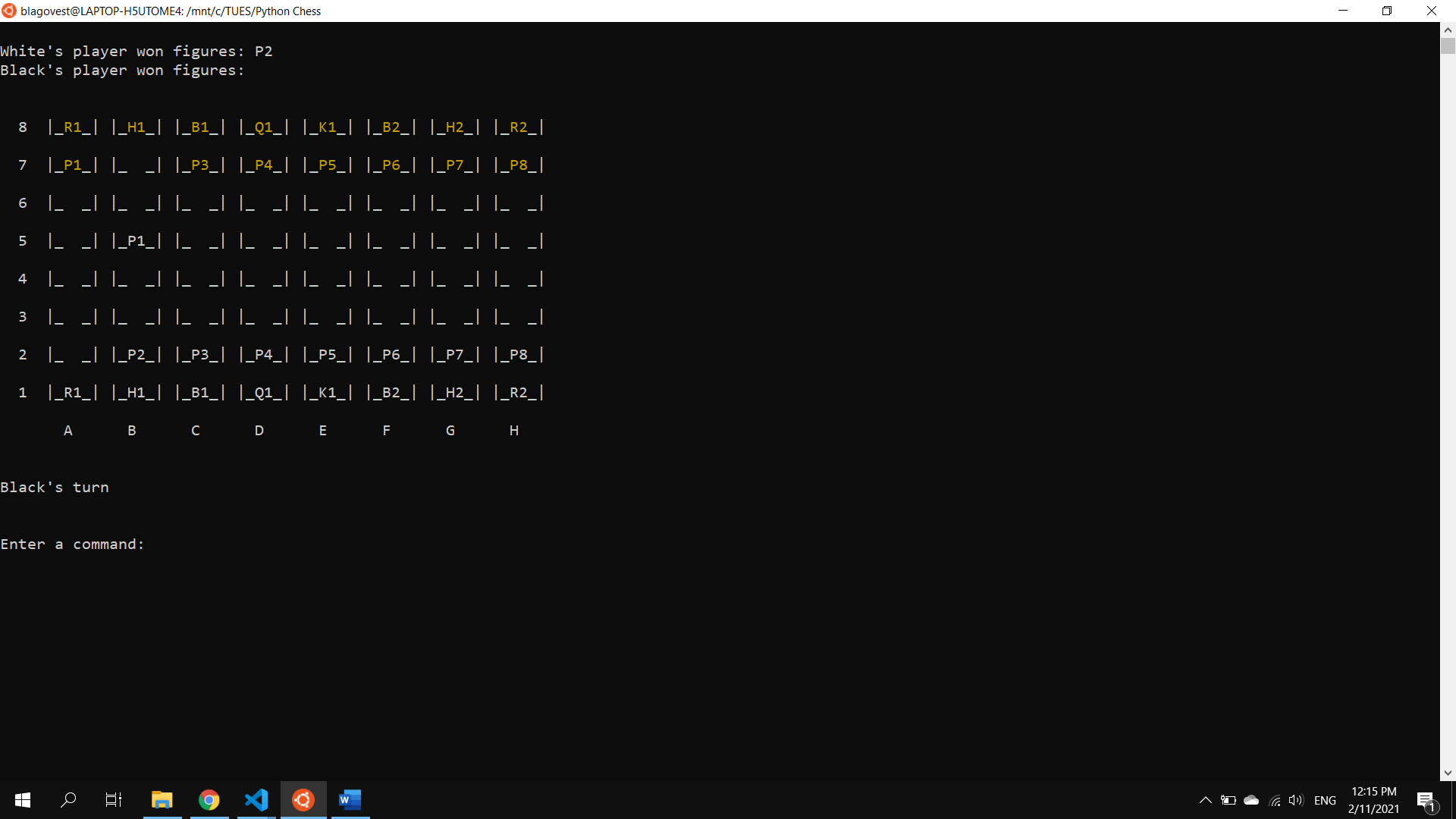
* създава се обект *Game*, в който ще се инициализира всичко
* вътре в Game обекта се създават:
  + дъска *ChessBoard*
  + 32 фигури *Figure* (бели + черни)
  + двама играчи *Player*, които взимат в себе си по 16 фигури

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* обекта.

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 „А2-А4“, 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\_rook( )* – тества движението и способността за взимане на фигури на пешка (тя е единствената Фиг., 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

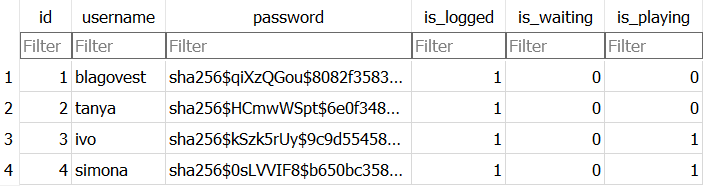
За да бъде възможно създаването на игра от външни ресурси (от променливи в тестовете) конструкторът на Game обекта приема няколко аргумента:

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

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

За да бъде възможно тестването на ходове в играта метода run на Game обекта 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.

**3.4. 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:

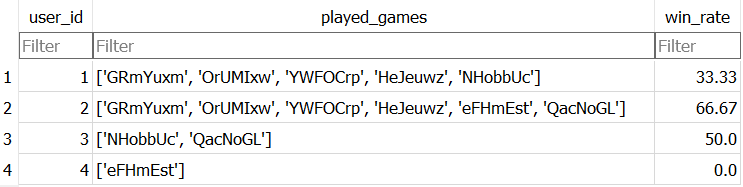
Лични атрибути:

* *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 за взаимно-изключващи се.

****

**Табл. 3.2.** Статистики за потребителите

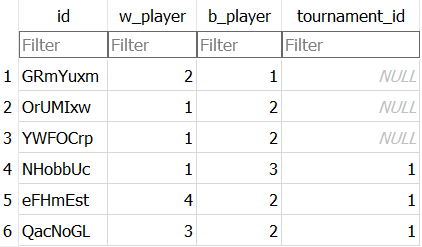
Втората таблица (Табл. 3.2.) е допълнение на първата и се казва „user\_stats“.

В нея се запазват следните данни:

* *user\_id* – идентификационен номер на потребителя (външен ключ)
* *played\_games* – списък с идентификационните имена на всички игри, които потребителя 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 като е превърнат в символен низ, като при записване на нови данни в него той трябва да се върне в първоначалното състояние на масив, за да може да са възможни каквито и да е операции.

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

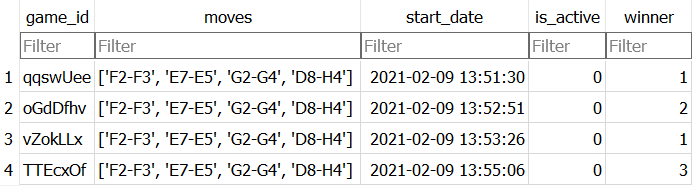
******

**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.

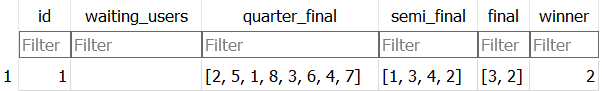
****

**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 чакащите биват изтрити)
* *quarter\_final* – списък с идентификационните номера на потребителите, започващи игра в четвърт финал
* *semi\_final* – списък с идентификационните номера на потребителите, започващи игра в полуфинал
* *final* – списък с идентификационните номера на потребителите, започващи игра в четвърт 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 съдържа 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, по време на която потребителят е изпратил съобщението
* *text –* текстът на самото съобщение
* *time –* датата и часа, в които съобщението е било изпратено

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

Сървърната част на приложението, написана на Python, 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 + профилната страница
* *game\_base* – отнася се конкретно към потребителската игра
* multiplayer – обслужва функционалностите, свързани с играта тип 1v1 (обикновен мултиплеър)
* *tournament* – насочен е към свързаните с турнир функционалности
* *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).

**3.5.1.***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 стринг със седем произволни букви (малки и големи), която се използва като идентификационно име за игра във функционалностите на другите файлове

**3.5.2.** *database.py* – създава базата данни и сесия към нея и съдържа един метод – *init\_db( )*, който създава метаданни за базата; от този файл се взима за всички други променливата за сесията на базата данни

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

- регистрация

- влизане и излизане от профил

- проверка дали потребителя е влязъл в профил

**3.5.4.** *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 такъв клас, който създава Python играта)

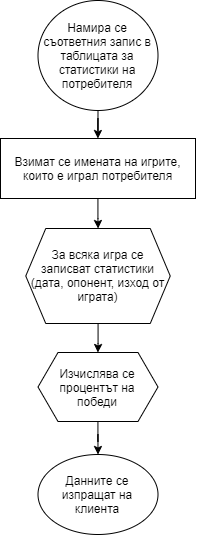
- *gameDetails* – допълва класа за играта

- *Tournament* – създава турнир

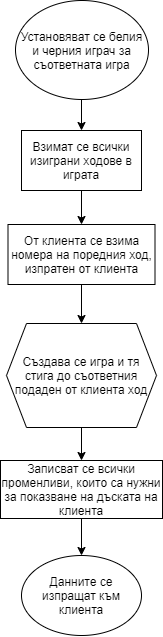
- *Message* – създава съобщение

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

**3.5.5.** *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

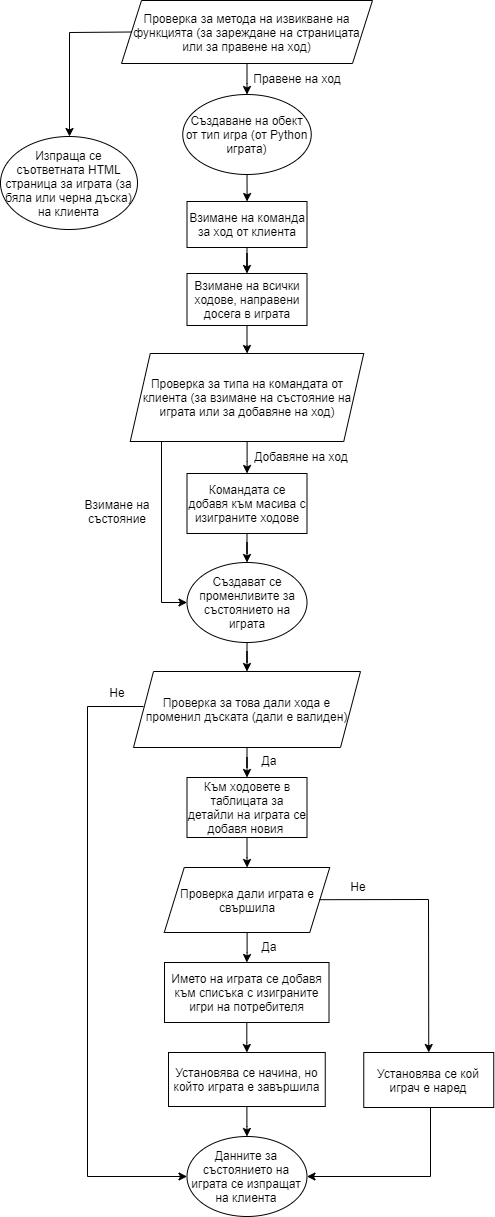


**Фиг. 3.4.** Страница за преглеждане на изиграни игри

**3.5.6.** *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

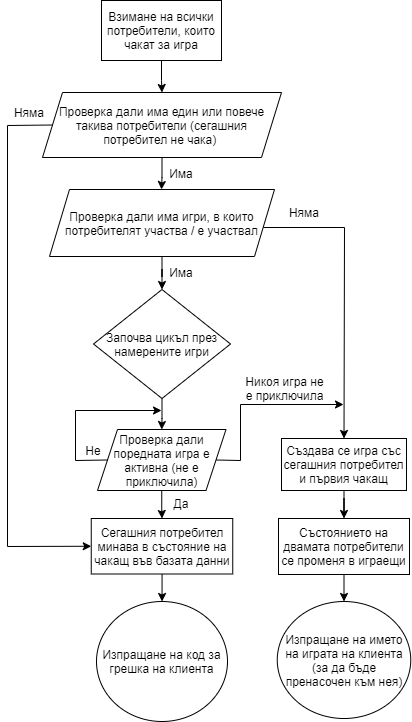
****

**Фиг. 3.6.** Протичане на игра

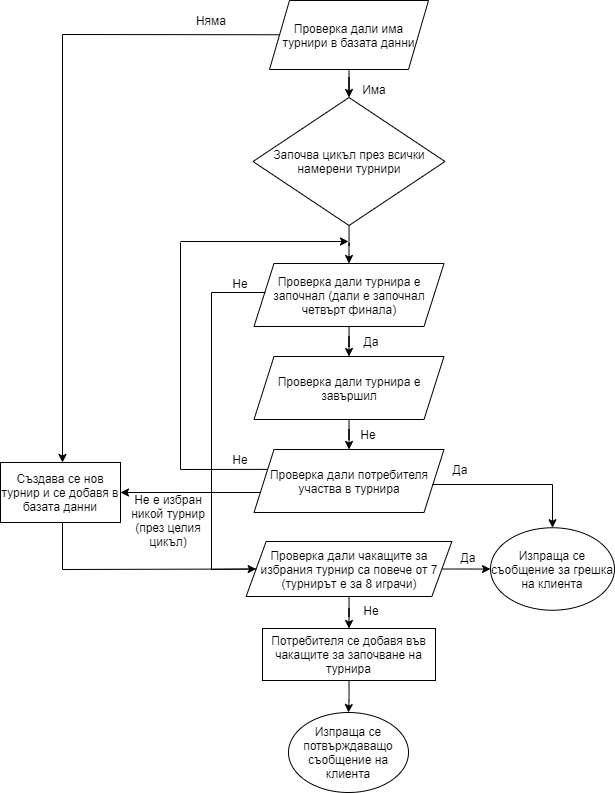
**3.5.7.** *multiplayer.py* – разгледани функционалности – влизане в игра (Фиг. 3.7.) и създаване на игра (Фиг. 3.8.).

****

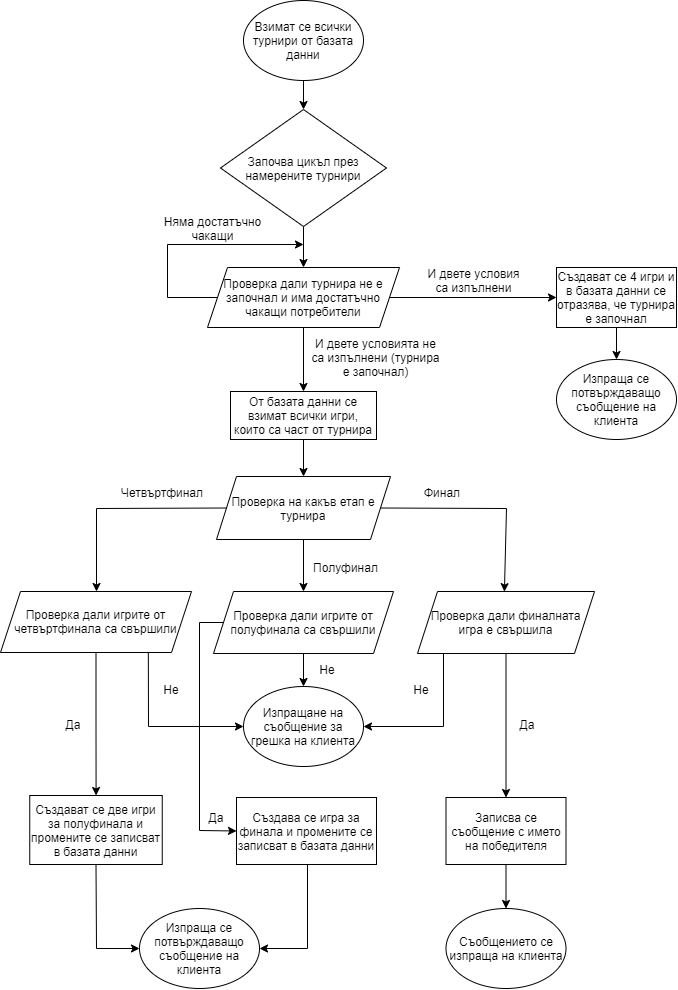
**Фиг. 3.7.** Влизане в игра

****

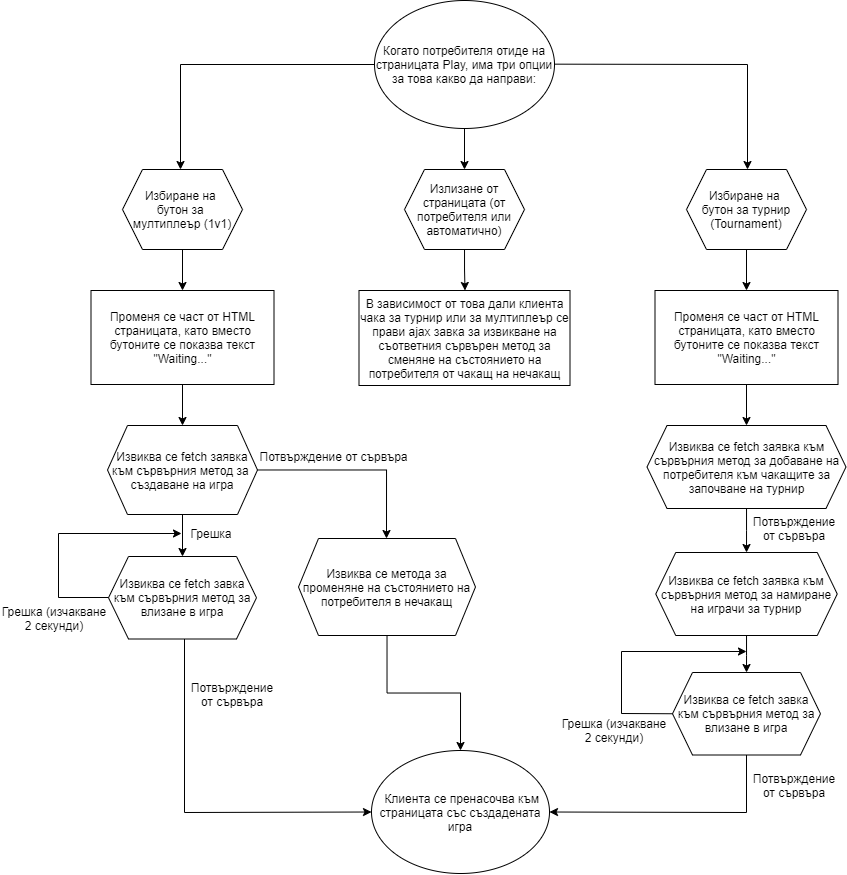
**Фиг. 3.8.** Влизане в игра

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

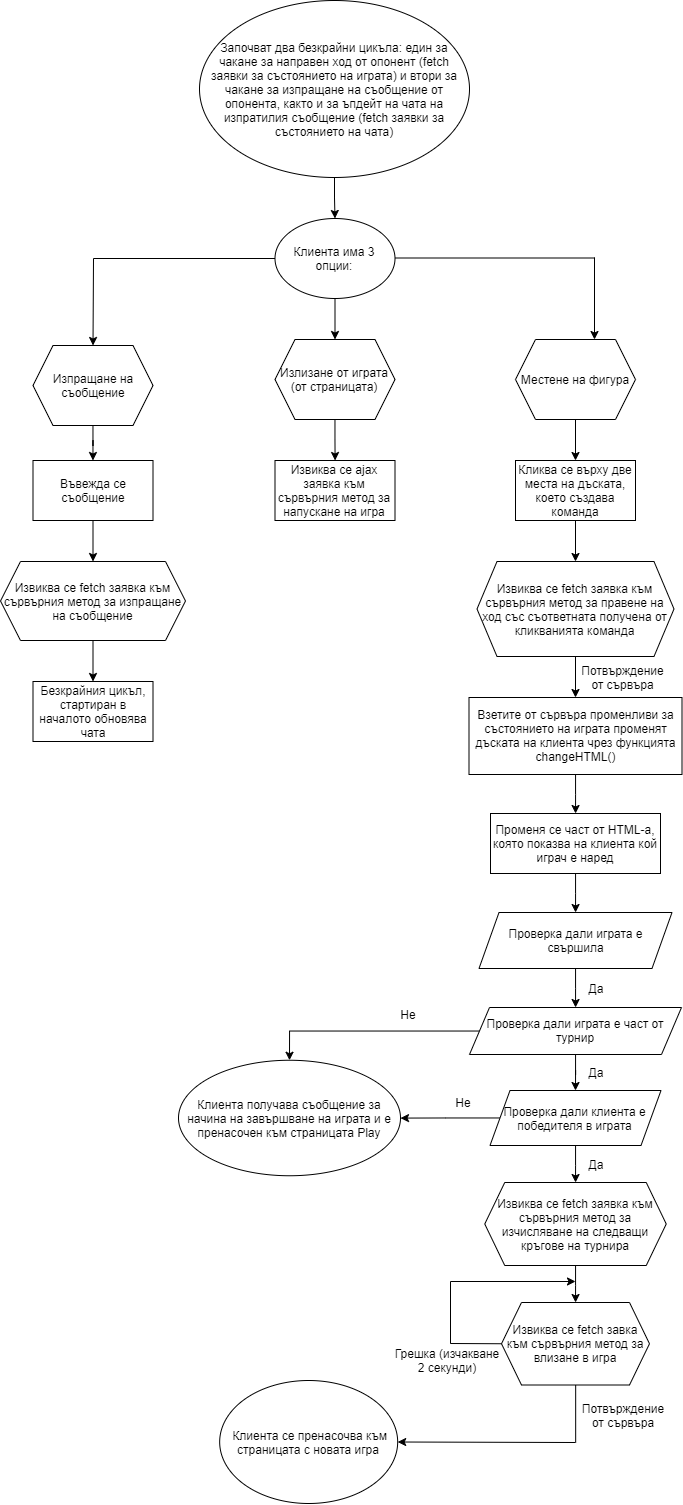
**Фиг. 3.9.** Намиране на играчи за турнир

****

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

**3.5.9.** 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 ще бъдат 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”

**ЧЕТВЪРТА ГЛАВА**

**РЪКОВОДСТВО ЗА ПОТРЕБИТЕЛЯ**

Това ръководство е ориентирано главно към потребителите на някаква форма на Linux операционна система (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 ситемата 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:

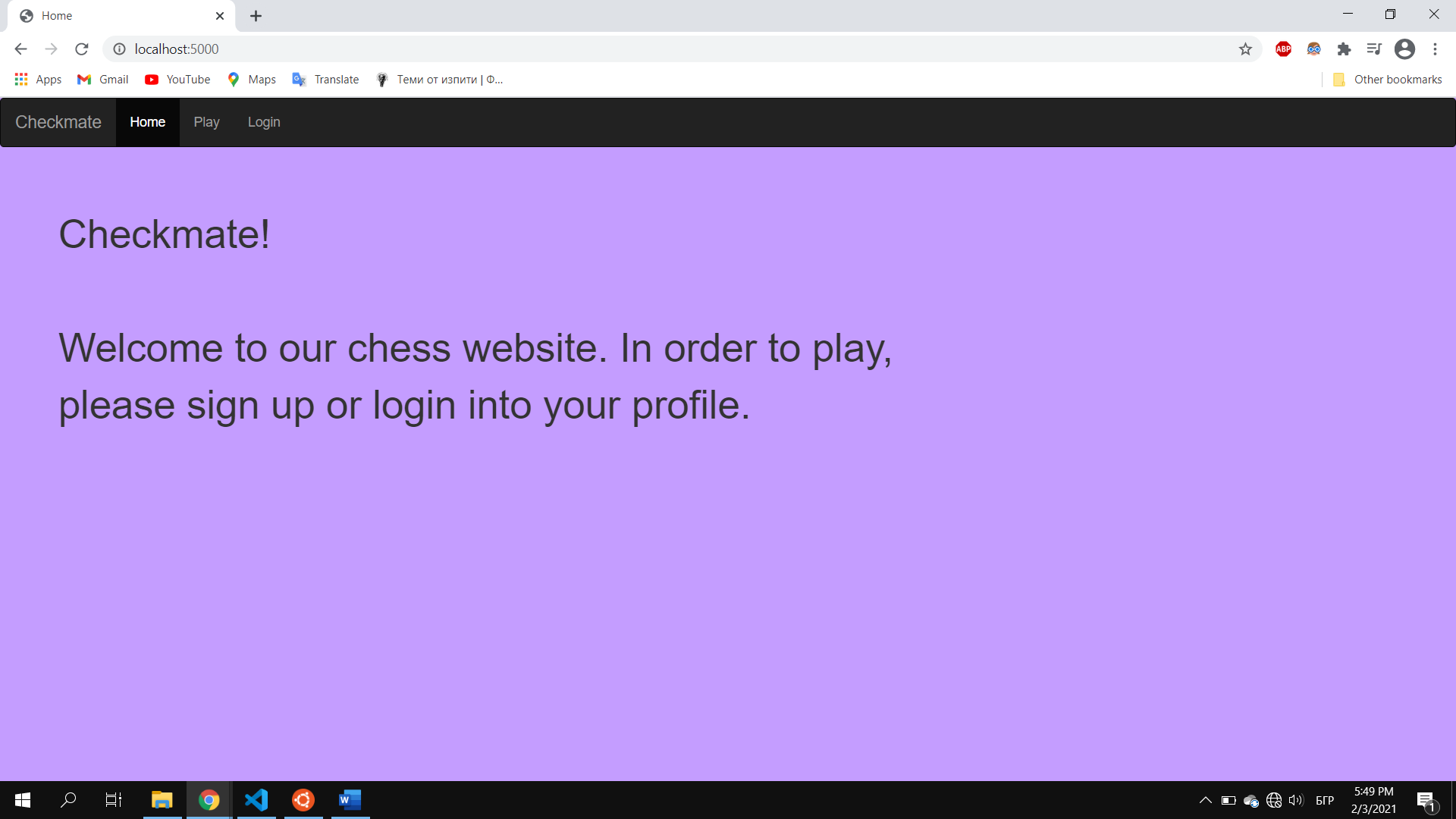
$ source ~/.bashrc

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

$ export FLASK\_APP=main.py

Сега можете да стартирате приложението (всеки път) с командата:

$ flask run

Ако искате да терминирате приложението, използвайте Ctrl+C.

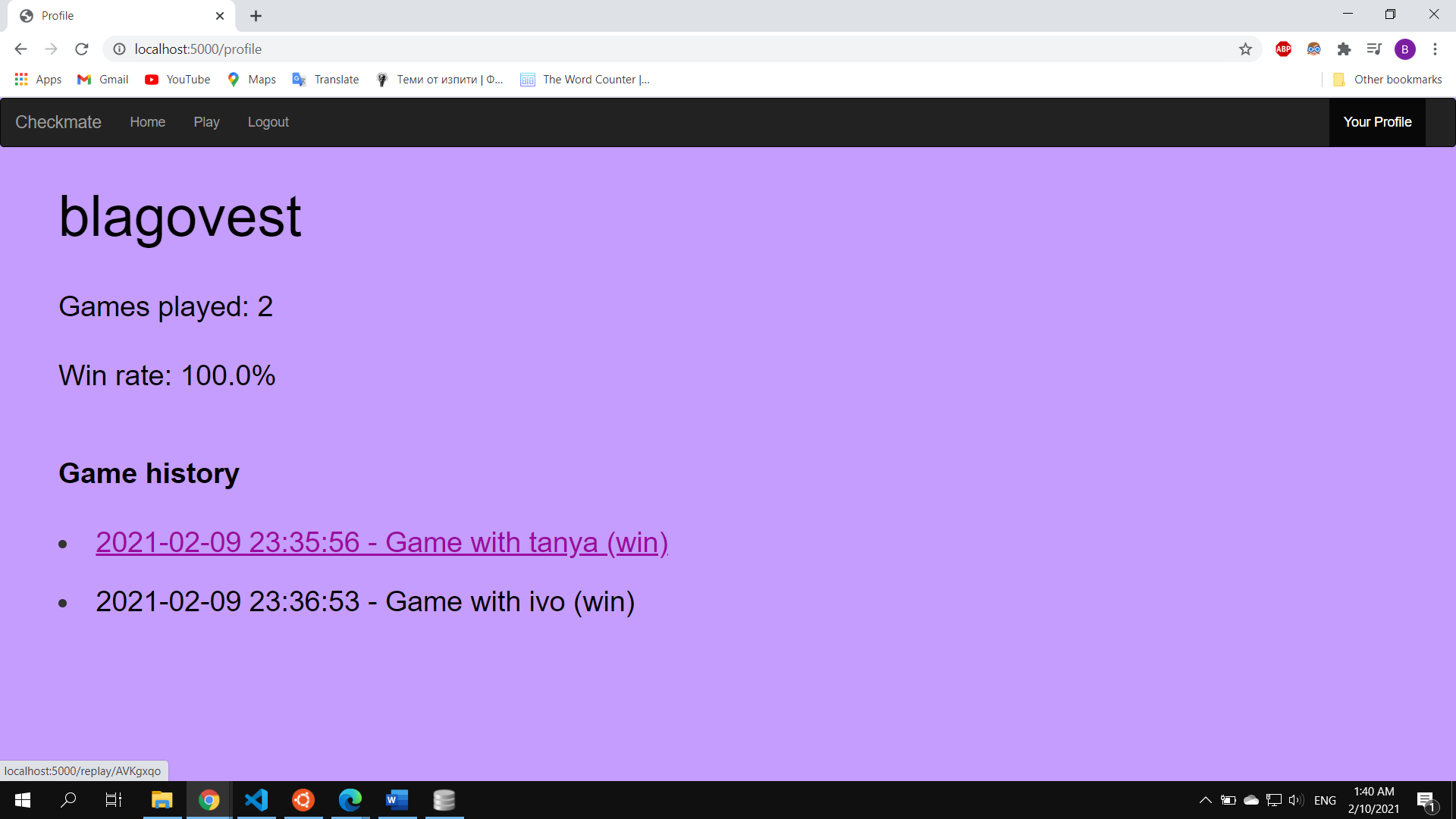
**Фиг. 4.1.** Начална страница

Щом пуснете приложението, заредете адреса на “localhost” в браузъра си 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 „Sign up“ линка 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.

Докато не сте влезли в профила си страницата за игра „Play“ ще ви пренасочва към „Login“. Щом влезете в профила си на менюто за навигация в най-лявата част ще се появи бутон „Your Profile“. Тези функции контролират свободата на потребителите в сайта и го правят по-интуитивен.



**Фиг. 4.2.** Профилна страница

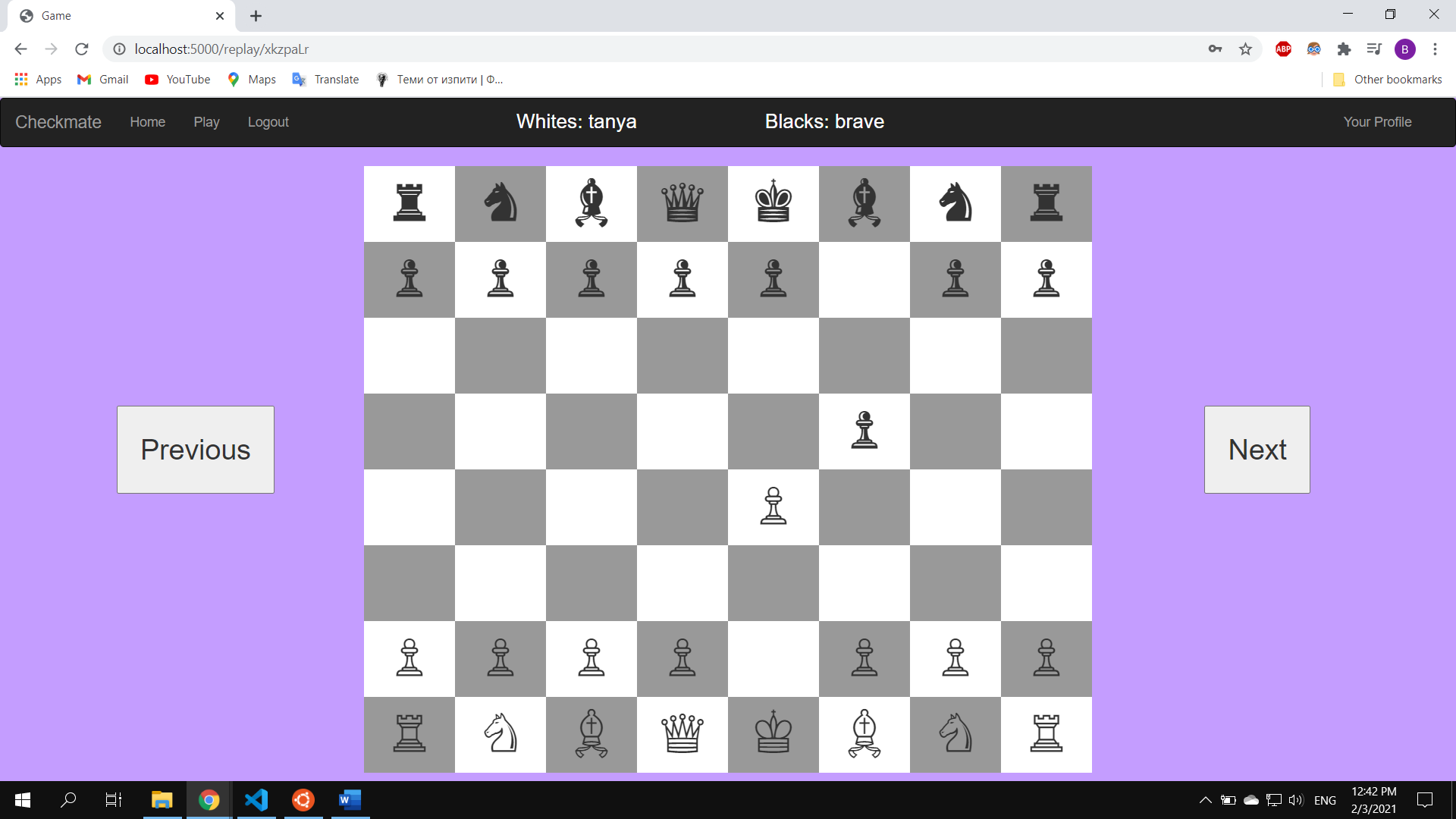
Профилната страница (Фиг. 4.2.) се състои от:

* името на потребителя
* брой на изиграните игри
* процент на спечелените игри
* история на игрите

Историята на игрите 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 равен.

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

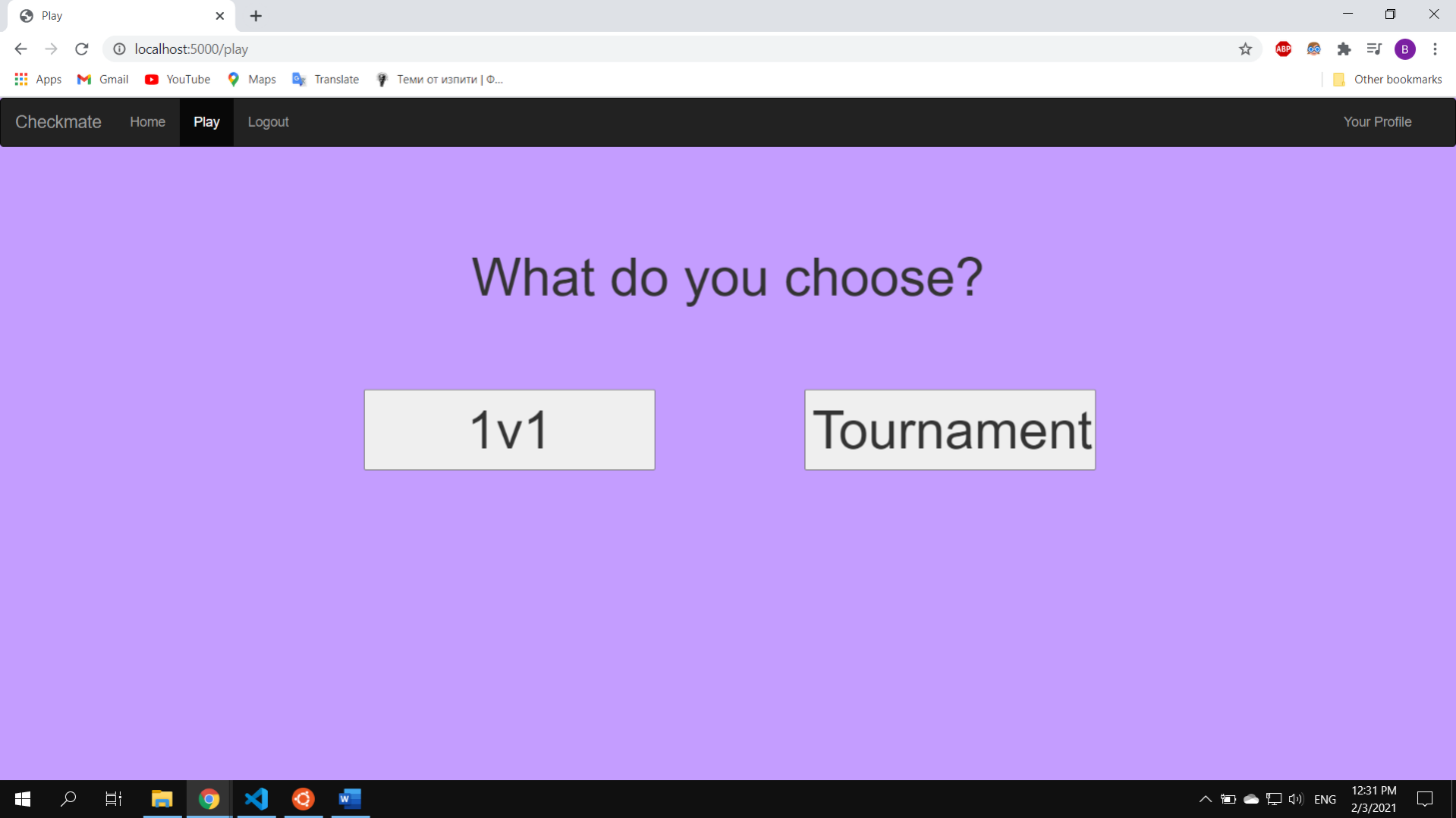


**Фиг. 4.3.** Страница за разглеждане на изиграна игра

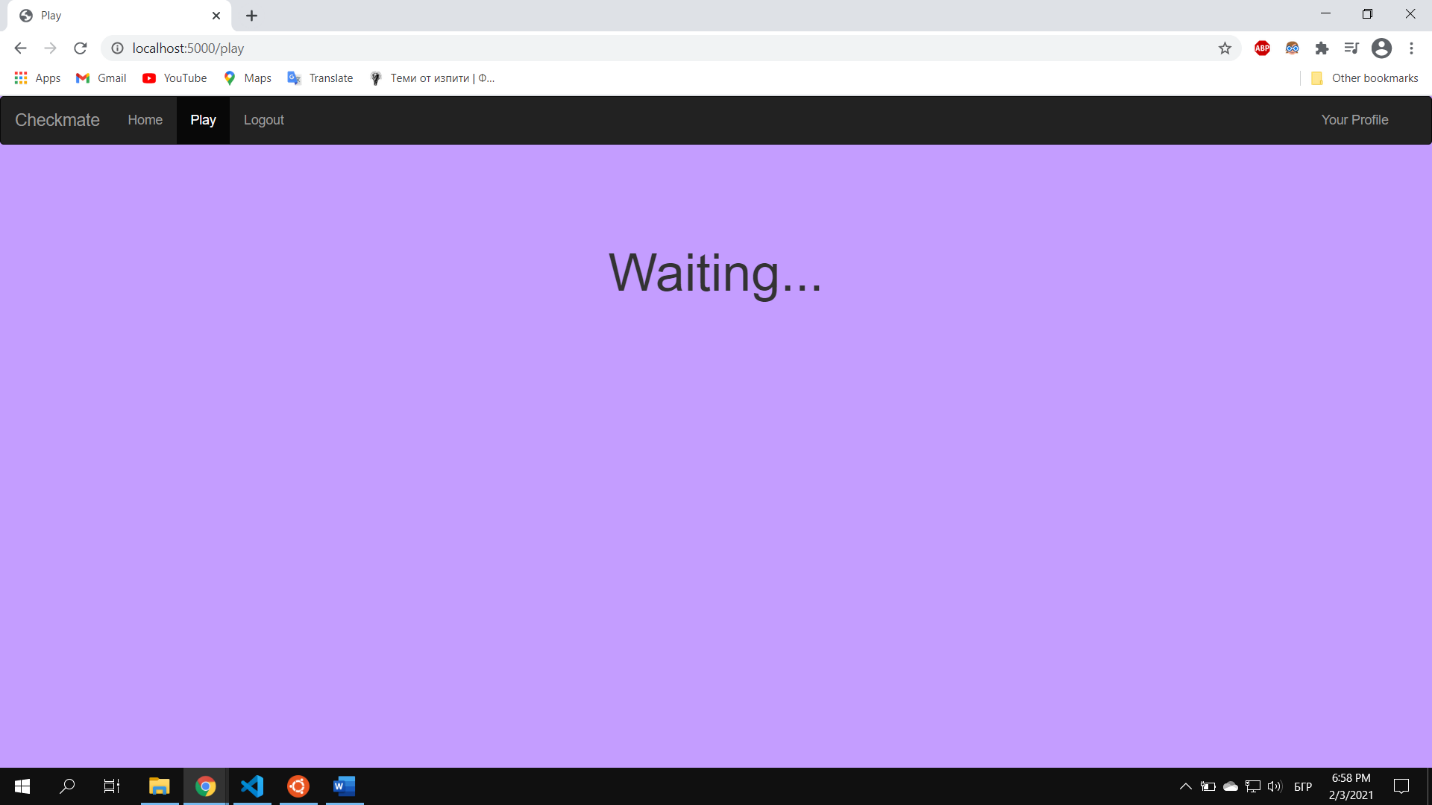
Над дъската (Фиг. 4.3.) пише кой играч с кои фигури е, и бутоните от двете страни преместват фигурите по ходовете, които са изиграни по време на играта.

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

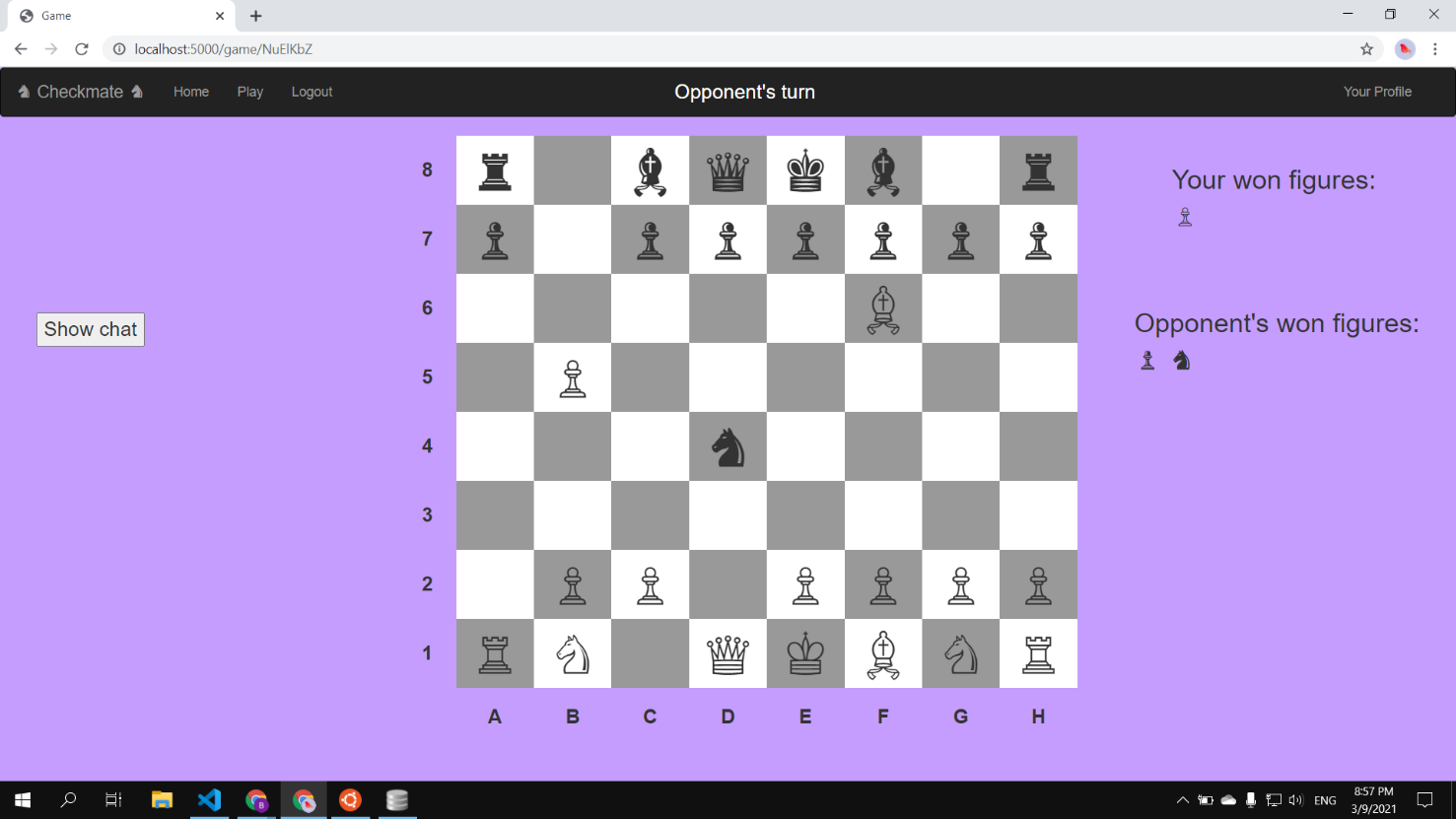
На страницата, която се зарежда (Фиг. 4.4.) имате две опции – да играете в нормален мултиплеър или в турнир.

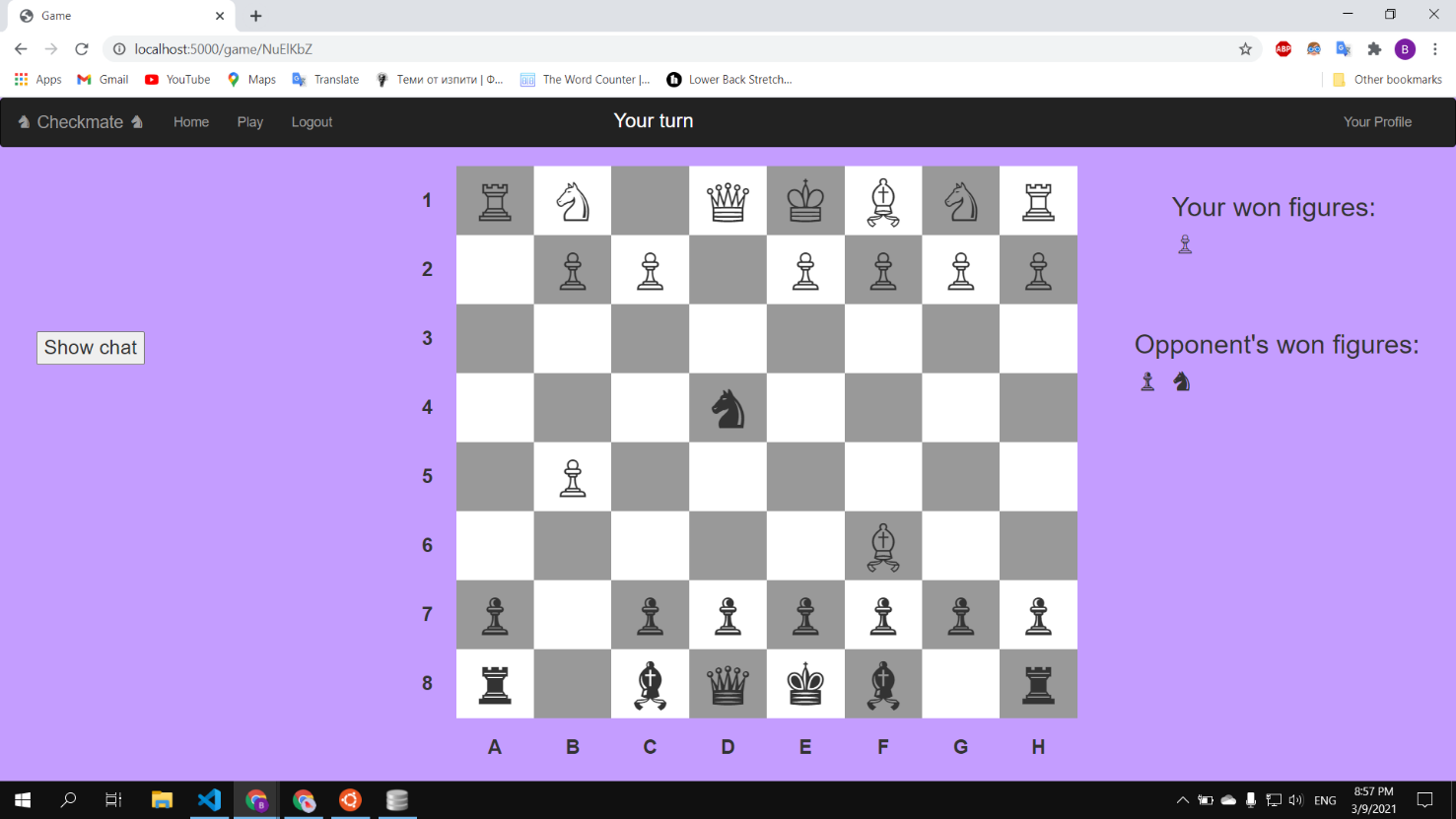
****

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

Избирайки една от опциите влизате в списъка за чакащите играчи за съответния тип игра.

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

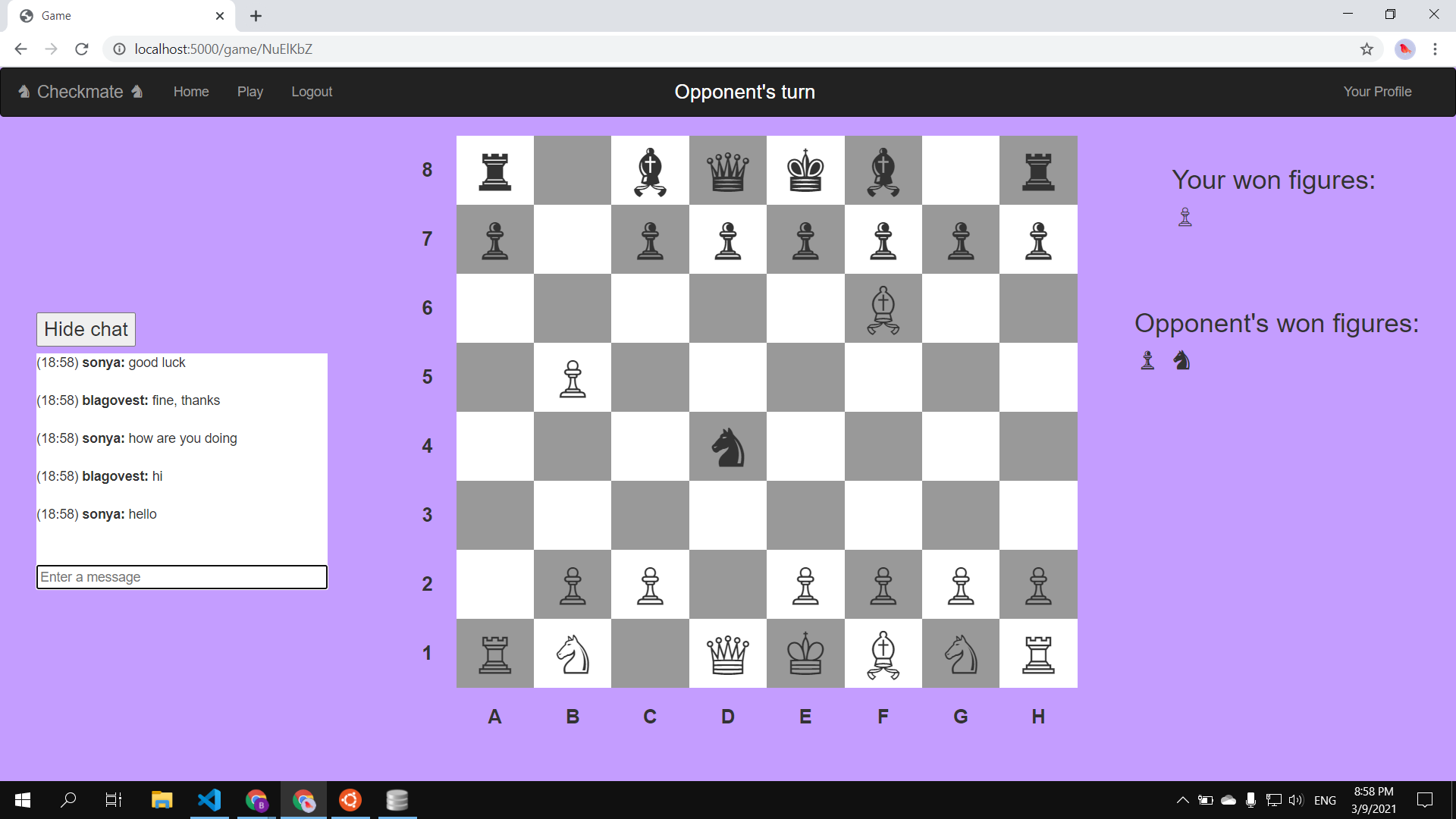
Нормалният мултиплеър започва при двама чакащи, а турнирът – при осем.



**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
* името на потребителя, който е изпратил съобщението
* текста 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 сървъра 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 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 използвани от системата
* да се имплементира някакъв вид комуникация между потребителите извън конкретна игра, например отбори от играчи и възможност за покана за игра между тях
* добра 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. “chess.com” - <https://www.chess.com/>
20. “lichess.org” - <https://lichess.org/>
21. SQLite - <https://sqlite.org/docs.html>
22. SQLAlchemy - <https://docs.sqlalchemy.org/en/13/>
23. Bootstrap - <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
24. Visual Studio Code - <https://code.visualstudio.com/docs>
25. Ubuntu WSL 2 - <https://docs.microsoft.com/en-us/windows/wsl/>
26. Git - <https://git-scm.com/doc>

**СЪДЪРЖАНИЕ**

1. Увод (стр. 3)
2. Първа глава – Методи и технологии за реализиране на WEB приложения
   1. Основни принципи, технологии и развойни среди за реализиране на WEB приложения (стр. 4)
   2. Съществуващи решения и реализации (стр. 6)
3. Втора глава – Проектиране на структурата на WEB приложение за игра на шах
   1. Функционални изисквания към WEB приложение за игра на шах (стр. 9)
   2. Съображения за избор на програмни средства и развойната среда (стр. 10)
   3. Проектиране на структурата на базата от данни (стр. 11)
4. Трета глава – Програмна реализация на WEB приложение за игра на шах
   1. Цялостна структура на приложението (стр. 15)
   2. Ядрото на проекта – играта на шах (стр. 16)
   3. Същност и детайли на WEB приложението (стр. 20)
5. Четвърта глава – Ръководство на потребителя (стр. 32)
6. Заключение (стр. 41)
7. Използвана литература (стр. 42)