

Софийски университет “св. Климент Охридски”

Факултет по Математика и Информатика

Специалност Инфомационни системи

Проект “Brick Breaker”

изготвили

**Виктор Драганов** ФН:71499

**Васил Пачеджиев** ФН:71486

курс IV

по избираема дисциплина „Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js“

Дата: 03:07:17

# Разширено описание

Проектът представлява онлайн игра за двама играчи. Потребителите се регистрират, след което използват акаунта си за достъп до системата. Името, с което потребителят се е регистрирал е уникално и му служи за да бъде намерен от други играчи. Системата предоставя информация дали въведеното име е вече използвано. При науспешен вход в системата се връща информация дали грешката е потребителското име или в паролата.

Игра може да бъде създадена чрез търсачката на други играчи или чрез избиране на някой свободен играч от списък в началната страница. След потвърждение на поканата на домакина, двамата играчи се пренасочват към екрана на играта. Там могат да си изпращат съобщения чрез чат. Посредством написването на командата START в чата от страна на домакина играта започва. Играчите използват мишките си за да местят палка, с която да удрят топчето. При неуцелване топчето от стана на някой играч той губи живот. При загуба на всички животи играчът губи играта. На полето се намират и тухли. При контакт на топчето с тухла тя се разрушава и последният ударил топчето печели точка. При събиране на определен брой точки играчът печели животи, с което увеличава шанса си за печалба.  
С течение на играта бързината на топчето се забързва, с което се увеличава и трудността и за двамата играчи.

В края на играта се отчитат печалбите и загубите. Тази информация определеля рейтинга на всеки играч. Колкото по-голяма част изиграните игри на даден играч са печалби, толкова по-висок е рейтингът му.  
  
Приложението има примитивен звуков ефект от ударите на топчето, както и музикален контролер с радио. В процеса на използване потребителят вижда различни модални диалози, прозорци или зареждащи индикатори, които да подсказват какво е текущото състояние на приложението.

# Изполвани технологии и библиотеки

Сървърната част е базирана на Node.js интерпретатора с Express.js server. Информацията за потребителите се складира в документ базирана база от данни Mongo.

Библиотеката използвана за връзка на сървъра с базата е mongoose. Сървърът предоставя статична директория(„/“), в която се намира клиентската част от приложението.

Клиентската част е имплементирана с библиотеката React.js. За рутиране из приложението служи библиотеката React-Router-Redux, в която се държи и основната информация служеща за работата на клиентската част. Дизайнът е на material-ui като са използвани готови компоненти имплементирани по стиловите правила на Google Material UI.

Сървърния и клиентския код са имплементирани на статично типизиран JavaScript. Използван е flow static type checker.

# Описание на не-тривиалните аспекти на архитектурата и реализацията на системата

Клиетът и сървърът си комуникират по два начина. Когато клиентът иска да извести сървърът например за регистрация/вход или се нуждае от дадена информация намираща се в базата това се случва посредством REST API, като сървърът използва body-parser middleware, за да обработи заявките.   
Когато клиентът иска да нотифицира друг клиент например за покана за игра, чат и по време на самата игра се изполва web socket протокола (socke.io).

# Описание на архитектурно значимите интерфейси в реализираната система и в частност подробно описание на REST service API и протокола за неговото използване

Route – user

* register
  + type - POST,
  + request body - {name: string, password: string}
  + response – the save user
* login
  + type - POST,
  + request body - {name: string, password: string}
  + response – 404 no such user, 202 successfully logged, 400 invalid password
* isLogged
  + type – GET
  + response – 401 or 200
* logout
  + type – GET
  + response – 400 or 200
* search
  + type – GET
  + request parameters – {query: string}
  + response – 400 Search query should be between 3 and 10 symbols, 404 no session, 200 JSON – Array<{name: string, available: boolean}>
* setSocket
  + type – POST
  + request body – {socketId: string}
  + response – 403 no session, 200
* usersOnline
  + type – POST
  + response – 500 for database error,

200 JSON – Array<{name: string, coefficient: number}>

Route – game

* playerCoefficient
  + type – GET
  + request parameters – {name: string}
  + response – JSON data will be a number in the range between zero and one and minus one if the player has not played a single game yet

# Описание на начина на инсталиране и конфигуриране на разработената система

В package.json файлът се намират скриптове за всяка нужна операция в прокта.

За разработка:

* npm run mongo – стартира базата от данни
* npm run start – стартира сървърът на nodemon интерпретатор, който от своя страна използва интерпретаторът babel-node, който е нужен за типизирания код. nodemon от своя страна следи за промени по кода на сървърната част и рестартира сървърът след всяко запазване на промяна.
* npm run watchClient – стартира webpack watcher който пакерита клиентската част с помощта на babel транспилатор след което следи за промени из файловете

За общо използване:

* npm run mongo – стартира базата от данни
* npm run buildServer – генерира сървърната част от кода без типовите анотации
* npm run serve – стартира Node.js интерпретатора с генерирания сървърен код
* npm run buildClient – генерира клиентския код просредством webpack and babel

# Заключeние

Бъдещото развитие на системата е да бъде по стабилна от страна на аутентикация и работа със сокет протокола. Физиката в играта да бъде заменена с библиотека създадена за целта. Да се събира още информация за изиграните игри с цел по добра статистика.