**ДЪРЖАВЕН ИЗПИТ**

**ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

**Д И П Л О М Е Н П Р О Е К Т**

Тема: „Разработване на уеб базирано приложение за огранизиране на спортни събития“

Практическа част:................................................

Дипломант: Николай Цветанов Стефанов, 12Б клас

професия: 481030 „Приложен програмист“

специалност: 4810301 „Приложен програмиране“

Ръководител на дипломния проект: инж. Любица Любенова Димитрова

Дипломант:.................

(подпис)

Pъководител:...............

(*подпис*)

София, 2025г.

Съдържание

[1. Въведение 3](#__RefHeading___Toc2604_3886614992)

[1.1. Задание за разработка 3](#__RefHeading___Toc2606_3886614992)

[1.2. Анализ на заданието 3](#__RefHeading___Toc2608_3886614992)

[1.3. Поставени цели 4](#__RefHeading___Toc2610_3886614992)

[2. Основна част 5](#__RefHeading___Toc2911_3562064950)

[2.1. Избор на технологии 5](#__RefHeading___Toc2913_3562064950)

[2.1.1 Избор на технология за потребителската част 5](#__RefHeading___Toc2915_3562064950)

[2.1.2 Избор на технология за вървъра 6](#__RefHeading___Toc3795_3886614992)

[2.1.3 Избор на технология за базата данни 8](#__RefHeading___Toc3797_3886614992)

[2.2. Софтуерен дизайн 10](#__RefHeading___Toc4011_3886614992)

[2.2.1 Етапи на разработка 10](#__RefHeading___Toc2921_3562064950)

[2.2.2 Софтуерна архитектура 12](#__RefHeading___Toc2923_3562064950)

[2.2.3 Описание и взаимодействие с приложението 13](#__RefHeading___Toc2925_3562064950)

[2.2.4 Графичен дизайн и структура на приложението 16](#__RefHeading___Toc2927_3562064950)

[2.2.5 Дизайн на сървъра 18](#__RefHeading___Toc2929_3562064950)

[2.2.6 Дизайн на базата данни 19](#__RefHeading___Toc3166_3562064950)

[2.2.7 CRUD заявки 27](#__RefHeading___Toc2931_3562064950)

[2.3. Тестване 33](#__RefHeading___Toc2933_3562064950)

[3. Заключение 34](#__RefHeading___Toc2935_3562064950)

[3.1. Постигнати цели 34](#__RefHeading___Toc2937_3562064950)

[3.2. Потенциално развитие 35](#__RefHeading___Toc2939_3562064950)

[4. Източници 36](#__RefHeading___Toc1876_2043827372)

[5. Приложение 37](#__RefHeading___Toc1878_2043827372)

# Въведение

## **Задание за разработка**

В съвременния свят спортните събития играят ключова роля в обществото, като осигуряват възможности за физическо развитие, социални контакти и професионална изява. Организацията на спортни турнири и състезания изисква ефективна система за регистрация и управление на участниците. Настоящият дипломен проект има за цел разработването на онлайн платформа, която улеснява процеса на създаване, организиране и управление на спортни събития.

Martial arts competition е уеб базирано проложение, което е разработено с помоща на основни технологии, а имено HTML, CSS и Javascript за фронт-енд, Node.js и Express.js като сървърна технология и не на последно място MariaDB за управление и използване на бази данни.

Уеб приложението предлага на потребителите си лесен, универсален и независим начин за регистране или ораганизиране на спортно събитие. С увеличаване на организираните състезания нуждата от подходяща технология е голяма, а до момента такава няма. Съществуващите на пазара са сложни и тежки за повечето устройства, поради многото анимации и неоптимизирани функции.

Платформата ще предоставя възможност за:

* Регистрация на потребители (състезатели, треньори и отбори).
* Създаване и управление на спортни събития.
* Записване за участие в събития.
* Управление на категории и генериране на състезателни списъци.

## Анализ на заданието

В приложението потребителите имат възможността да създават нормален потребителски акаунт, който има частичен достъп до функционалностите на сайта. След това този потребителски акаунт може да се превърне в такъв със статус треньор или състезател.

Неоторизиран потребител може само да разглежда и изтегля основните файлове за състезанията, като жребии и времеви графици. За да участва в състезание, е нужно предварително да се регистрира през уеб платформата. Самата регистрация е направена да бъде лесна и интуитивна.

Събира се основна информация за потребителя (име, фамилия, телефон, електронна поща, парола, дата на раждане и пол, както и снимка, която не е задължителна). След което се създава записа и се автентикира. Потребителите имат правата да променят и персонализират акаунтите си.

Потребител със стандартен профил има възможност да извърши регистрация за събития със статус „свободно за всички" (free for all). Тези състезателни прояви са с редуциран мащаб и не налагат задължителна клубна регистрация – оптимален вариант за индивиди с непрофесионален интерес към спортната дисциплина, желаещи първоначално участие в състезателна активност.  
 Потребителски профил със статус „треньор" се присъжда на лице, което след първоначална регистрация и валидация на представлявания отбор, след удостоверяване на автентичност на необходимите документи, придобива разширени административни правомощия. Този тип акаунт се характеризира с най-висока степен на системни привилегии, включващи функционалности за създаване и администриране на събития. Акаунтът позволява публикуване и актуализиране на жребий и времеви график в информационния раздел на съответното събитие, където информацията е достъпна за преглед от други потребители. В допълнение, профилът осъществява мениджмънт на представлявания отбор, като определя състезателите, регистрирани като членове на съответния клуб.

Потребителски профил със статус „състезател" се характеризира с разширени възможности, които освен регистрация за събития със свободен достъп, включват и правото на участие в клубни състезания (team events).

Основната функционалност на системата е генерирането на оповестителен интерфейс за спортно събитие. Представител с треньорски правомощия има възможност да конфигурира състезателни категории, специфични регламенти, максимален количествен лимит на участници в категория, графичен идентификатор, географска локация и статус на състезателното събитие. Процесът е проектиран с оптимална опростеност, същевременно предоставяйки на организатора инструментариум за адаптиране на събитието към спецификите на съответната бойна дисциплина.

## **Поставени цели**

Основната цел на проекта е разработването на стабилна, сигурна и интуитивна платформа, предназначена за фасилитиране на организационните и изпълнителните процеси, свързани със спортни събития. За реализирането на тази стратегическа цел са формулирани следните оперативни задачи:

* ****Създаване на система за регистрация и вход**** – осигуряване на възможност за създаване на профили, влизане в системата и управление на лични данни.
* ****Разработване на модул за създаване и управление на събития –** позволяващ на**

**организаторите да добавят събития, да задават категории и да следят участниците.**

* Изграждане на функционалност за записване на участници – даваща възможност

на състезатели и треньори да се регистреират за събититя.

* **Проектиране на удобен потребителски интерфейс** – с интуитивен дизайн и лесна

навигация.

* **Внедряване на база данни** – за сигурно съхранение на всички потребителски данни,

събития и резултати.

* **Разработване на функция за генериране на списъци** – експортиране на CSV

файлове с участници по категории за екипа, отговорен за кантара.

* **Провеждане на тестове и оптимизации** – за гарантиране на функционалност, сигурност и стабилност на платформата.

# **Основна част**

## **Избор на технологии**

### **Избор на технология за потребителската част**

**HTML** (HyperText Markup Language) формира основната структура на уеб страниците в проекта, дефинирайки елементите и тяхната йерархия. Като стандартен език за уеб документи, HTML осигурява семантично правилно представяне на съдържанието, което е от ключово значение за достъпността и SEO оптимизацията.

**CSS** (Cascading Style Sheets) се използва за придаване на визуален стил и оформление на HTML елементите в проекта. Той позволява отделяне на дизайна от структурата на

страницата, което улеснява поддръжката и промените в оформлението без засягане на съдържанието.

**JavaScript** (клиентска част) е основният програмен език от страна на клиента, позволяващ динамично взаимодействие с потребителя без презареждане на страницата. Той обработва събития, изпраща AJAX заявки към сървъра и манипулира DOM дървото, създавайки по-интерактивно потребителско изживяване. Използването на JavaScript и от двете страни (клиент и сървър) позволява споделяне на код и по-ефективна разработка.

**EJS** (Embedded JavaScript) е шаблонна система, която вмъква JavaScript директно в HTML, позволявайки генериране на динамично съдържание на сървъра. Тя е избрана вместо по-сложни фронтенд фреймуърци поради простотата, минималната крива на обучение и близостта до стандартния HTML. EJS позволява лесно предаване и използване на данни от сървъра, което е идеално за приложение със смесено рендиране, където част от страницата се генерира динамично, а друга част се манипулира чрез клиентски JavaScript.

**Bootstrap 5** e избран като CSS фреймуърк поради богатата колекция от готови компоненти и класове, които ускоряват разработката на потребителския интерфейс. Той осигурява консистентен, мобилно-отзивчив дизайн с минимални усилия и намалява нуждата от писане на персонализиран CSS. Bootstrap е особено полезен за този проект, тъй като позволява бързо създаване на форми, таблици и навигационни елементи, които са основни компоненти в системата за управление на състезания.

### **Избор на технология за вървъра**

**Node.js** е избран като сървърна платформа за проекта поради способността му да изпълнява JavaScript на сървъра, което позволява използването на един език за целия стек на приложението. Неговата неблокираща, асинхронна природа е идеална за обработка на множество едновременни заявки без да се блокира основната нишка, което е от съществено значение при управление на състезания с голям брой участници и регистрации. Богатата екосистема от npm пакети осигурява готови решения за множество функционалности, спестявайки значително време при разработката. Node.js осигурява много добра производителност при I/O операции, което е от ключово значение при запитвания към база данни и файлова система. Благодарение на събитийно-ориентираната архитектура, Node.js е особено ефективен при приложения с реално време, каквито са елементи от този проект като актуализиране на турнирни схеми и регистрации.

**Express.js** е избран като уеб фреймуърк, надграждащ Node.js, поради неговата простота, гъвкавост и минималистичен подход към разработката на уеб приложения. Той осигурява интуитивен API за маршрутизиране на HTTP заявки, което значително улеснява дефинирането на пътища и крайни точки за различните модули в проекта (събития, клубове, регистрации). Мощната middleware архитектура на Express позволява лесно добавяне на функционалности като автентикация, сесии и обработка на файлове чрез обединяване на модули, разработени от общността. Express се интегрира безпроблемно с множество шаблонни системи, включително EJS, която се използва в проекта за генериране на динамично HTML съдържание. За разлика от по-тежките фреймуърци, Express не налага строга архитектура, което дава свобода в структурирането на проекта според специфичните нужди.

**Bcrypt** е избран за хеширане на пароли в проекта поради своята висока устойчивост срещу атаки с груба сила и речник. За разлика от по-прости хеш алгоритми, bcrypt автоматично добавя уникална сол (salt) към всеки паролен хеш, което значително повишава сигурността дори при изтичане на информация от базата данни. Алгоритъмът позволява настройка на фактора на работа, което дава възможност за балансиране между сигурност и производителност според нуждите на проекта. Bcrypt е проектиран специално за сигурно съхранение на пароли и е стандарт в индустрията, с доказана ефективност през годините.

**Express-session** предоставя удобен механизъм за управление на потребителски сесии в приложения, базирани на Express. Библиотеката се грижи за създаването на уникални сесийни идентификатори и съхраняването на сесийни данни, по подразбиране в паметта на сървъра, но с възможност за използване на различни хранилища. Express-session автоматично създава и управлява бисквитки за проследяване на сесиите, като поддържа множество опции за сигурност, включително срок на валидност, ограничения по път и домейн. В проекта express-session се използва за съхранение на информация за влезлия потребител, което позволява персонализиране на съдържанието и контрол на достъпа до различните функционалности на платформата.

**Multer** е middleware за Express, специализирано в обработката на multipart/form-data заявки, което го прави идеален за управление на качването на файлове в уеб приложения. В проекта се използва за обработка на профилни снимки, документи на клубове и банери на събития, като предоставя прецизен контрол върху дестинацията, именуването и ограниченията на качваните файлове. Multer работи безпроблемно с Express, интегрирайки се в middleware веригата и автоматично добавяйки информация за файловете към обекта на заявката. Библиотеката поддържа валидация по тип, размер и други критерии, което помага

за предотвратяване на злонамерени файлове и поддържане на консистентност в съхраняваните данни.

**PDFKit** е библиотека за програмно генериране на PDF документи директно от Node.js код. В проекта се използва за създаване на турнирни схеми, сертификати и други официални документи, свързани със състезанията. Библиотеката предлага богат API за прецизен контрол върху страниците, текста, шрифтовете, изображенията и векторната графика, което позволява създаване на сложни и визуално атрактивни документи. PDFKit поддържа работа с потоци, което прави генерирането на PDF ефективно дори за големи документи. Генерираните PDF файлове са стандартизирани и могат да се отварят с всеки PDF четец, което осигурява широка съвместимост за участниците и организаторите на събитията.

**JWT** библиотеката предоставя инструменти за създаване и проверка на JSON Web Tokens, които се използват за сигурно предаване на информация между страните в JSON формат. В проекта JWT се използва за реализиране на stateless автентикация при API заявките, където не е удобно или ефективно да се използват сесии. Токените съдържат информация за потребителя, кодирана и подписана с таен ключ, което позволява на сървъра да верифицира самоличността без допълнителни заявки към базата данни. JWT подпомага мащабируемостта на системата, тъй като не изисква съхранение на състояние на сървъра и позволява лесно хоризонтално скалиране. Библиотеката осигурява опции за срок на валидност на токените и поддържа различни алгоритми за подписване, което предлага гъвкавост в сигурността според нуждите на проекта.

### **Избор на технология за базата данни**

В основата на всяко съвременно уеб приложение стои базата данни – технология, която дава възможност за структурирано съхранение, обработка и извличане на данни. Базите данни осигуряват надеждно управление на информацията, поддържайки нейната интегритет, достъпност и сигурност. Проектът за управление на състезания се нуждае от система, която може надеждно да обработва сложни релации между участници, клубове, събития и регистрации, осигурявайки същевременно бърз достъп и обработка на данните при интензивно използване. MariaDB е избрана за технологя за бази данни за този прект.

**Предимства на MariaDB**

MariaDB е избрана като система за управление на бази данни в проекта поради няколко ключови предимства. На първо място, тя е напълно безплатен софтуер с отворен код

(FOSS - Free and Open Source Software), което премахва лицензионните разходи и позволява пълна прозрачност на кода. Това е съществено предимство пред комерсиални алтернативи като Oracle или MS SQL Server, особено за проекти с ограничен бюджет. MariaDB е създадена от оригиналните разработчици на MySQL като негов истински форк, запазвайки пълна съвместимост с MySQL, но същевременно предлагайки множество подобрения в производителността и сигурността.

**Сравнение с алтернативни решения**

В сравнение с PostgreSQL, друга популярна FOSS база данни, MariaDB предлага по-лесна настройка и поддръжка, което е предимство за проекти със средна сложност като нашата система за управление на състезания. Докато PostgreSQL може да предложи по-мощни функции за сложни заявки и типове данни, MariaDB предоставя отлична производителност за четене, което е критично за този проект с множество заявки за визуализиране на събития, участници и резултати.

За разлика от NoSQL решения като MongoDB или Cassandra, MariaDB осигурява строга структура на данните и гарантирана консистентност чрез релационния модел и поддръжката на ACID транзакции. Това е решаващо за приложение, където връзките между ентитите (клубове, членове, събития, категории, регистрации) са сложни и изискват интегритет на данните.

Спрямо SQLite, който е подходящ за вградени или малки приложения, MariaDB предлага значително по-добра скалируемост, едновременен достъп от множество потребители и стабилност при по-голямо натоварване, което е необходимо за уеб приложение с потенциално множество регистрации по време на събития.

**Специфични предимства на MariaDB за проекта**

MariaDB внедрява иновативни функции по-бързо от MySQL, включително подобрени storage engines, оптимизации на заявки и функции за сигурност. Нейните разширени възможности за индексиране са особено полезни при търсене в големи набори от данни, като списъци с участници или исторически резултати от събития.

Инструментите за репликация на MariaDB позволяват лесно внедряване на архитектури с висока достъпност, което е важно за приложение, където недостъпността по време на състезание може да има сериозни последствия. Вградената поддръжка за Galera Cluster осигурява възможност за хоризонтално скалиране при бъдещо разрастване на системата.

В контекста на нашия проект, вградените функции на MariaDB за сигурност, включително напреднала система за привилегии и опции за шифроване, помагат за защита на личните данни на участниците, което е от първостепенно значение за спазване на регулации като GDPR.

От досега проучената информация, избахме MariaDB, понеже е втората най-предпочитана технология в сферата на уеб разработването и уеб програмирането. Информацията се запазва в строго определен ред, което намалява шанса за грешки.

## **Софтуерен дизайн**

### **Етапи на разработка**

Има много стъпки, включени в процеса на разработка на уеб приложения. И количеството работа, извършена на всяка стъпка, зависи от обхвата на проекта. Нашата визия, бюджет, график, изисквания към персонала и технологични изисквания също играят фактор. Най-общо казано обаче процесът на разработка на уеб приложение включва следните основни стъпки:

**1. Събиране на информация и поставяне на основни цели**

**2. Планиране**

**3. Дизайн на уеб приложение**

**4. Кодиране и програмиране**

**5. Копирайтинг и етикетиране**

**6. Тестване, преглед и стартиране**

**7. Поддръжка и актуализации след стартиране**

1. **Събиране на информация**

Този етап е изключително важен и представлява началната фаза на проекта. Екипът провежда задълбочено проучване за нуждите на клиента, целевата аудитория и пазарните изисквания. Извършва се анализ на съществуващи решения, конкуренти и потенциални предизвикателства. Провеждат се интервюта с крайните потребители, събират се детайлни изисквания и очаквания. Целта е да се създаде ясна и подробна картина за това какво точно цели да постигне уеб приложението, какви функционалности да включва и какви проблеми да решава.

1. **Планиране**

На този етап се трансформира събраната информация в конкретен план за действие. Дефинират се точни цели, обхват на проекта, и ключови изисквания. Създава се подробен проектен план, който включва разпределение на ресурси, определяне на екипни роли, изготвяне на график и бюджет. Идентифицират се потенциални рискове и се разработват стратегии за тяхното управление. Избират се подходящите технологии, programming езици и инструменти, които ще бъдат използвани по време на разработката.

1. **Дизайн на уеб приложение**

Този етап включва създаването на цялостната визуална и функционална концепция на приложението. Дизайнерите разработват wireframes, mokup-и и прототипи, които визуализират бъдещия потребителски интерфейс. Създава се детайлен дизайн, който включва цветова схема, типография, разположение на елементите и цялостно потребителско изживяване. Извършва се проектиране на архитектурата на приложението, включително структура на данните, навигация и интеракции между различните компоненти.

1. **Кодиране и програмиране**

По време на този етап разработчиците превръщат дизайна и плановете в реален работещ код. Създават се front-end и back-end компоненти на приложението, имплементират се необходимите функционалности, бази данни и интеграции. Следват се предварително определени стандарти за кодиране, използват се системи за контрол на версиите като Git. Паралелно с писането на код се извършва вътрешен преглед и оптимизация на кода, като се цели постигане на най-добра производителност и качество.

1. **Копирайтинг и етикетиране**

Копирайтингът и етикирането са критични комуникативни елементи в процеса на разработка, които осигуряват ефективното представяне и разбиране на продукта или услугата. Копирайтингът помага да се създаде убедително послание, което обяснява ценността на продукта, привлича вниманието на потенциалните клиенти и комуникира ключовите негови характеристики. Етикирането гарантира, че продуктът отговаря на всички законови и регулаторни изисквания, предоставяйки прозрачна и точна информация за крайните потребители. Двата процеса работят заедно, за да изградят доверие, да информират и да направят продукта по-разбираем и атрактивен за пазара. Те са същностни елементи от маркетинговата стратегия, които подпомагат успешното позициониране и приемане на продукта от целевата аудитория.

1. **Тестване, преглед и стартиране**

Този критичен етап включва щателна проверка на всички аспекти на разработеното приложение. Извършват се различни видове тестване: функционално тестване, тестване на производителност, сигурност, съвместимост с различни устройства и браузъри. Целта е да се открият и отстранят всички възможни грешки и да се гарантира, че приложението напълно отговаря на първоначалните изисквания. След успешното тестване се извършва официалното пускане на приложението, което включва качване в производствена среда, първоначална конфигурация и миграция на данни.

1. **Поддръжка и актуализация след стартиране**

Заключителният етап включва постоянно наблюдение и подобряване на приложението след неговото пускане. Екипът събира и анализира обратна връзка от потребителите, следи за евентуални технически проблеми и бързо ги отстранява. Извършват се редовни актуализации за подобряване на сигурността, оптимизация на производителността и добавяне на нови функционалности. Целта е да се гарантира дългосрочната ефективност и актуалност на уеб приложението съобразно променящите се нужди на потребителите и технологичните тенденции.

### **Софтуерна архитектура**

Архитектурният модел "Model-View-Controller" (MVC) е избран като основен структурен подход в разработката на настоящия проект. Този модел представлява утвърден метод за организация на софтуерни приложения, който осигурява ясно разделение на отговорностите между компонентите, повишава поддръжката на кода и улеснява паралелната разработка. MVC архитектурата разделя приложението на три взаимосвързани, но логически обособени компонента – модел (данни и бизнес логика), изглед (потребителски интерфейс) и контролер (обработка на заявки и координация), което позволява по-структуриран и модулен подход към разработката на софтуера.

**1. Moдел (Model)-** Моделът може да се разглежда като интелектуалното ядро на всяко софтуерно приложение, представляващ централната логика за управление на данни и бизнес процеси. Той съхранява цялата вътрешна информация, която приложението обработва, като включва структури за данни, правила за валидация, методи за съхранение и извличане на информация, и основните бизнес логики, които определят как данните се създават, четат, актуализират и изтриват. В контекста на нашото уеб приложение, моделът би съхранявал информация за събития, категории, регистрации, потребителски профили и всички вътрешни правила за обработка на регистрациите. Независимостта на модела от потребителския интерфейс позволява преизползването му в различни части на приложението или дори в напълно различни приложения, което го прави изключително гъвкав и адаптивен.

**2. Изглед (View)-** Изгледът представлява визуалната презентационна част на приложението, отговорна за представянето на данните по начин, който е интуитивен и достъпен за крайния потребител. Той трансформира суровите данни, получени от модела, в структурирани и естетически приятни интерфейси, които могат да включват HTML страници, мобилни екрани, графични елементи или друга форма на потребителско представяне. Основната роля на изгледа е да комуникира информацията по възможно най-четливия и разбираем начин, като същевременно осигурява интерактивни елементи за взаимодействие с потребителя. При уеб приложенията, изгледът би включвал шаблони за страниците, формуляри за въвеждане на данни, таблици с резултати, графики и всички визуални елементи, които потребителят вижда и с които взаимодейства.

**3. Контролер (Controller**)- Контролерът действа като интелигентен посредник, свързващ модела и изгледа, управлявайки цялата логика на взаимодействие между тях. Той прихваща входящите заявки от потребителя, обработва ги, комуникира с модела за извличане или модифициране на данни и избира подходящия изглед за представяне. Можем да го сравним с мениджър, който получава инструкции, взема решения, извършва необходимите действия и докладва резултатите. При уеб приложение контролерът ще обработва заявки като натискане на бутон, попълване на формуляр или навигация между различни секции, като координира цялостното поведение на приложението. Неговата способност да разкачва различните компоненти позволява по-голяма гъвкавост и по-лесна поддръжка на цялостната система.

MVC моделът предоставя редица съществени предимства за софтуерната разработка. Чрез строгото разделение на отговорностите между компонентите, той позволява паралелна работа на различни специалисти - дизайнери могат да променят интерфейса, без да засягат бизнес логиката, докато програмисти могат да усъвършенстват вътрешната функционалност, без да се притесняват за визуалното представяне. Тази модулност значително опростява процесите на разработка, тестване и поддръжка, като намалява риска от взаимно влияние между различните части на приложението.

### **Описание и взаимодействие с приложението**

При достъп до уебсайта, потребителят се насочва към началната страница, където ясно се представят целта и функциите на приложението. Навигационното меню в горната част на страницата е интуитивно и предоставя бърз достъп до основните раздели – преглед на събития, вход и регистрация.

Нерегистрираните потребители имат възможност да разглеждат всички налични събития, да изтеглят PDF файлове с жребии и планови графици, както и да преглеждат списъците с участници в отделните категории, без обаче да могат да се записват за участие в събитията.

Модулът за събития представя карти с ключова информация - банер или лого, наименование, периоди за регистрация и провеждане, както и локация. Над картите е разположена търсачка, която осигурява бързо и лесно намиране на желаните състезания.

За записване в състезание или създаване на събитие е необходима регистрация в системата. Това изискване предотвратява претоварването с ненужни събития и фиктивни регистрации, които биха могли да нарушат нормалното функциониране на платформата. Треньорите могат да публикуват обяви за събития единствено след официално потвърждение и валидиране на данните за съответния клуб.

След регистрация потребителят придобива възможност да се записва за първенства, да разглежда и се присъединява към клубове. Записването за участие се извършва от детайлната страница на събитието, към която потребителят се пренасочва от страницата с обяви. На тази страница участникът може да избере категории, отговарящи на неговата възраст и пол.

Категориите са представени чрез интерактивни карти, които променят цвета си според броя записани състезатели: син цвят означава наличие на свободни места, оранжев показва, че категорията е почти запълнена, а червен цвят сигнализира, че категорията е пълна и затворена за нови регистрации.

В горната част на страницата, близо до основната информация за събитието, се намира динамичен елемент, който автоматично променя цвета си и заключва възможността за нови регистрации в зависимост от датата и часа. Тези функционалности, заедно с опциите за изтегляне на план-график и жребий, осигуряват на всички участници необходимата информация за успешната организация на мероприятието.

През профилната страница, потребителя може да променя личните си данни с

изключение на възраста, името и пола си, с цел запазване на интегритета на потребителя, както и функционалност за променяне на парола.

В основата на проекта стои функционалността за създаване и организиране на събития чрез разработената уеб платформа. Този процес се осъществява от потребители с треньорски привилегии. Създаването на обява става посредством попълване на опростена форма, която позволява на организатора да въведе цялата необходима информация за събитието, както и да дефинира различните състезателни категории и приложимите правила.

За допълнително управление е създадена страница за контрол на събитието. Достъпът до нея се осъществява, когато собственикът на събитието отиде на страницата с обявите и се пренасочи към собственото му събитие. Страницата с административни права се отключва само когато идентификаторът на потребителя съвпада с този на създателя на събитието, осигурявайки ексклузивен достъп само за организатора.

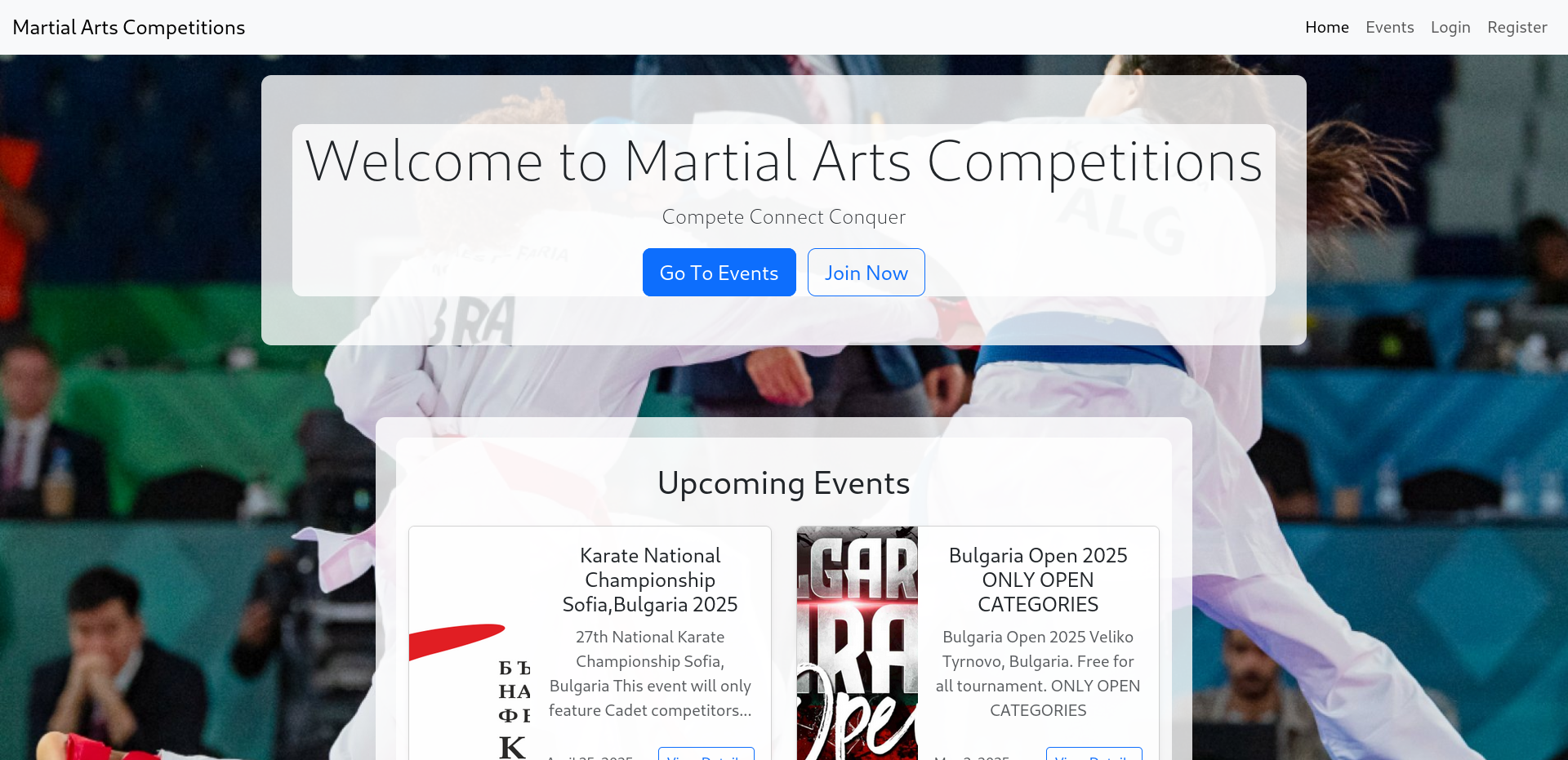
На тази страница са интегрирани функционалности за добавяне и премахване на категории, качване на жребии и план-график, изтегляне на списъци с участници по категории във формат CSV, одобряване и отхвърляне на състезатели в категориите, редактиране на детайлната информация и премахване на обявата. Тук също са видими интерактивните елементи, показващи броя на вече одобрените състезатели, както и бутонът за изтриване на събитието.  
 В секцията за регистрации организаторът вижда всички списъци с категории и заявките на състезателите за участие. В горната част е разположена търсачка, която позволява бързо намиране на конкретен списък при наличие на множество категории. Тези функционалности осигуряват пълен контрол на организатора, като същевременно правят работния процес максимално опростен и ефективен.

Последната важна функционалност е контролът и управлението на клуба. Това е критичен елемент за по-големите състезания, където регистрацията е ограничена само до клубни състезатели. Собственикът на клуба може да приема или отхвърля заявки за присъединяване и да изпраща покани към състезатели. След одобрение, потребителят получава статус на състезател за съответния клуб, което му дава право да се регистрира за състезания, изискващи клубно членство.

Всеки тип потребител има специфични права, като съзнателно не е създаден профил с абсолютно пълен достъп, за да се избегнат конфликти между различните организатори. Системата е проектирана така, че потребителите да създават и управляват обяви напълно

самостоятелно и безплатно, без непрекъсната нужда от административна намеса, която би забавила процеса.

### **Графичен дизайн и структура на приложението**

 ***фиг. 2.2.4.1 Начална страница***

Графичният дизайн започва от началната страница на приложението (виж фиг. 2.2.4.1). Уеб сайтът е разработен с фокус върху простотата и ефективността при работа. Използвани са синьо и бяло, което създава минималистичен и чист дизайн. Тези цветове помагат за бързо ориентиране. Сините елементи обикновено са бутони или важна информация, които се забелязват лесно, без да са натрапчиви или трудни за виждане от хора с по-слабо зрение. Тази цветова схема се използва в целия уебсайт за по-интуитивна навигация и по-добро потребителско изживяване.

На фигурата виждаме част от дизайна - съчетанието между цветовете, заоблените ръбове на картите и полупрозрачните рамки, които плавно свързват различните елементи. В горната част се намира менюто, достъпно от всички страници. Елементите в него са подредени според важността и честотата на използване, като същевременно позволяват на новите потребители да се регистрират. Навигационната лента се променя според правата на потребителя, но запазва основната си структура и само добавя функции, което осигурява познато и удобно изживяване.

За да имаме добре организирано приложение за събития, то се състои от няколко на брой страници:

* **Начална страница** (Index page виж: фиг. 2.2.4.1): Това е първата страница на уебсайта, през която всеки потребител преминава при влизане в приложението. Новодошлият потребител вижда най-скорошните предстоящи събития. Този детайл е изключително полезен, тъй като позволява да се пропусне търсенето в страницата със състезания и да се премине директно към регистрация или преглед на информация за конкретно събитие.
* **Страница със събития** (event page виж: фиг. 5.1): Това е основната страница на приложението, която позволява на потребителите да разглеждат всички събития - както стари, така и новосъздадени. В отделните карти се вижда кратка информация за състезанието, включително място, дати за регистрация и провеждане. Страницата е максимално опростена, с лесна за използване търсачка и възможност за сортиране по дати.
* **Страница за вход и регистрация** (login and register page): В тези страници потребителите влизат в профилите си с електронна поща и парола или създават нов потребителски акаунт. Формата е оптимизирана, а дизайнът запазва минималистичния и чист стил с белосините нюанси, което улеснява новите потребители.
* **Профинла страница** (profile page виж: фиг. 5.3): Профилната страница се състои от два главни модула. Първият позволява на потребителя да променя информацията по акаунта им. Вторият е за сменяне на паролата. В цветовата палитра също така се вкарва и сиво, което отделя пасивните прозорци и модули от белия фон.
* **Подробна страница за събитието** (event details page виж: фиг. 5.2): Това е персонализираната страница за събитието. Тук, за разлижа от малките кутийки на събититята, може да се види пълната информация за събитието отляво, както и банера вляво. Различните компоненти като категории, времеви график, жребии и локация са разделени в различни под прозорчета така, че да заемат едно място на екрана. Прескачането между тях става с натискането на бутона, чиято информация иска да се покаже. По същия начин се продължава със синьо-сивия нюанс, който присъства навсякъде, обединявайки страницата за събитята с тази по външен вид и дизайн.
* **Страниците за създаване на отбор и събитите** (create event, create club): Тези две страници са формуляри с еднаква структура. Те са подредени по ясен и логичен начин – както като дизайн, така и по разположение на полетата. Потребителят лесно и бързо попълва нужната информация, без излишни разсейвания или затруднения. Заглавието на формуляра е със син фон, за да се откроява по-добре и да бъде видимо и за хора с по-слабо зрение.
* **Страницата за редактиране и управление на събитите** (manage event page виж: фиг. 5.4): Страницата за контрол и редакция на събитие е най-голямата и сложна страница в целия проект. В нея главният дизайн клони повече към формите за попълване на информация от гледна точка на цветовата гама. Използва се повече синьо с целта да се взима напредвид важността на действията, които ще бъдат направени. В тази страница с червено са бутоните за изтриване на категория или на самото събитие. За управлението на категориите потребителят се препраща към под страницата на списъците със записалите се за събитието. В нея се използват всички различни цветове:
* Сиво - Отделя модулите и ги кара да изпъкват.
* Синьо – Лесно се вижда къде започват различните списъци за категориите.
* Зелено – Одобрените състезателите в списъците на категориите.
* Жълто – Чакащите за одобрение състезатели в списъците на категориите.
* Червено – За отхвърляне на одобрение. То може да е както от организатора на

събитието, така и от системата при запълнена категория или при опит за регистрация на не отговарящ на изискванията състезател.

* **Страница за управление и редакция на отбора** (manage club page): Тази страница следва добре познатия ни досега дизайн и цветово кодиране на графичния интерфейс. В страницата има 3 главни модула. Информативният модул показва информацията зададена от създателя. Статитстическият модул показва информация за клуба като състезатели и събития. Третият модул представлява набор от прозорци, които отговарят за одобряването на зявки и пращането на покани към други потребители. Всичките са в сиви рамки с които се открояват от фона, а бутоните са кодирани съответно със синьо и червено, като цвета им зависи от зададената им функционалност.

Дизайнът на всички страници е еднотипен и четлив. Базира се на мималистичната философия, на която е изграден графиченят интерфайс на страницата. Това предразполага по-лесното използване на стари устройства, като не затормозява машините компоненти с тежки анимации и ефекти.

### **Дизайн на сървъра**

Проектът е разработен с използването на Node.js и Express.js – технологии, които осигуряват бързина, гъвкавост и мащабируемост. Тази стек технологията се използва поради нейната способност да обработва множество заявки едновременно и богатата екосистема от библиотеки, които ускоряват процеса на разработка.

Когато потребител взаимодейства с приложението (например регистрация за събитие), процесът започва с HTTP заявка, която достига до Express.js сървъра. Express

действа като входна точка и маршрутизатор, приемайки заявката и насочвайки я към съответния обработващ модул според URL адреса и HTTP метода.

Body-parser влиза в действие още в началото на обработката, като анализира JSON или form-data съдържанието на заявката и го прави достъпно за контролерите чрез обекта req.body. Ако заявката включва качване на файл (например банер на събитие), Multer се намесва, за да обработи файловата част, съхрани файла в определена директория и добави метаданните му към заявката.

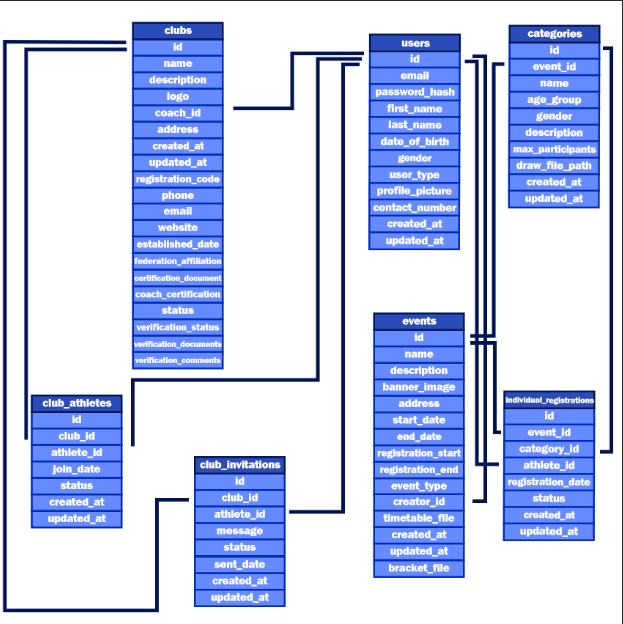
Express Sessions проверява бисквитките на заявката, за да идентифицира потребителската сесия. Ако съществува валидна сесия, данните за текущия потребител се прикачват към обекта на заявката, позволявайки контролерите да знаят кой извършва действието.

След тези предварителни стъпки, заявката достига до специфичния контролер (например eventsController или usersController). Контролерът съдържа бизнес логиката и използва модулите на приложението, за да извърши необходимите операции. Например, при регистрация за събитие, контролерът ще провери права, валидира данни и подготви заявка към базата данни.

Комуникацията с базата данни се осъществява чрез MySQL2 драйвера, който превръща JavaScript обекти в SQL заявки и ги изпраща към MariaDB. Драйверът използва асинхронни операции, така че сървърът не блокира докато чака отговор от базата данни. При регистрация на нов потребител, Bcrypt се използва за сигурно хеширане на паролата преди съхранение.

След получаване на данните от базата, контролерът обработва резултата и подготвя отговор. За HTML страници, EJS шаблоните се използват за генериране на динамично съдържание, комбинирайки статични шаблони с данни от сървъра. За API заявки, данните се форматират като JSON и се връщат директно.

### **Дизайн на базата данни**



***фиг. 2.2.6.1 Релационен модел на базата данни***

Таблицата **users** съхранява информация за всички потребители в системата. Тя включва основни лични данни като имейл, парола (хеширана за сигурност), име, фамилия, дата на раждане, пол и телефонен номер. Важен аспект е полето user\_type, което може да бъде "regular" (обикновен потребител), "coach" (треньор) или "athlete" (състезател). Това определя какви действия може да извършва потребителят в системата.

Полета в таблица **users:**

* **id** — Тип **int(11)**. Тази колона служи като уникален идентификатор за всеки

потребител, които служи като пъричен ключ. Това е най-важното поле в цялата, понеже е най-използваната референция.

* **email** — Тип **varchar(255)**. Основна колона съхраняваща електронната поща на

потребителя, използва се за вход, изпращане и получаване на покани за присъединяване към клуб.

* **password\_hash** — Tип **varchar(255)**. В тази колона се съхранява криптирана версия

на паролата под формата на стринг. Това е важна колона, поради критичната информация, която държи. Използва се при вход.

* **first\_name —** Тип **varchar(255)**. Тази колона съхранява първото име на потребителя.

Попълването на това поле е задължително понеже чрез него потребителят ще бъде разпознаван от други потребители на платформата, както и по време на състезание.

* **last\_name** – Тип **varchar(255)**. Тази колона съхранява фамилията на потребителя.

Попълването на това поле също е задължително понеже чрез фамилията си ще бъде разпознаван от други потребители, както и по време на състезание.

* **date\_of\_birth —** Тип **date**. Тази колона е отговорна за съхраняването на датата на

раждане на потребителя. Чрез датата за раждане се изчислява в коя възрастова група сътезателят може да се регистрира.

* **gender** — Тип **enum(„male“, „female“)**. Тази колона отговаря за съхранението на пола

на потребителя. Това е задължително поле, понеже чрез него се определя в кои категории клиентът може да участва.

* **user\_type** - Тип **enum(„regular“, „coach“, „athlete“)**. Тази колона отговаря за правата на потребителя. Спрямо него се извършват множество проверки и отключват функционалности. Автоматично се задава статус „normal“ при регистриране на потребителски акаунт.
* **profile\_picture** — Тип **varchar(255)**. Тази колона съхранява пътя към профилната

снимка на потребителя, която е качена на сървъра.

* **contact\_number —** Тип **varchar(20)**. Тази колона съхранява телефония номер за връзка

с потребителя. Забелязва се, че нарочно не е изолзван типът int, за да се избегнат излишни затруднения при записването на международни номера.

* **created\_at** — Тип **timestamp**. Tази колона отговаря за съхранението на времевия запис

на създаване на профила. Служи за спомагателно поле при отстраняване на грешки, както и да се пази за хронология.

* **updated\_at** - Тип **timestamp**. Тази колона стартира със същата стойност на created\_at, но се обновява при всяка промяна на потребителскят профил.

Таблицата **clubs** представлява спортните клубове. Всеки клуб има име, описание, лого, треньор (свързан с таблицата users), адрес, данни за контакт и информация за федерацията, към която принадлежи. Клубовете минават през процес на верификация, което се отразява в полето verification\_status.

Таблица  **clubs:**

* **id —** Тип int(11). Тази колона служи като уникален идентификатор за всеки клуб.
* **name** – Тип **varchar(255)**. Това поле съхранява името на клуба. Показва се в страницата на клуба, както и в списъка с клубове.
* **description —** Тип **text**. Тази колона отговаря за съхранението на допълнителна

информация за клуба, която да се покаже на клубната страница или при разглеждането на клубове.

* **logo —** Тип **varchar(255)**. Това поле съхранява пътя до снимката на клуба.
* **coach\_id** — Tип **int**. Тази колона служи за външен ключ към създателя на клуба (външен ключ към таблица users).
* **address –** Тип **varchar(255)**. Тази колона съхранява физическият адрес на клуба. Може да бъде променяна в бъдеще от създателя на клуба.
* **created\_at** и **updated\_at —** Тип **timestamp**. Тази колона съхранява времеви запис на

създаване и актуализиране на клубната информация.

* **registration\_code** — Тип **varchar(50)**. Колоната съхранява до 50 символен уникален код за регистрация, който се използва при присъединяване.
* **phone, email, website** — Тип **varchar** съответно **20**, **255**, **255**. Tези колони съхраняват

контактната информация за клуба.

* **established\_date —** Тип **date**. Тази колона съхранява дата на основаване на клуба.
* **federation\_affiliation** — Тип **varchar(255)**. Тази колона съхранява принадлежността на клуба към спортна федерация, което позволява бъдеща интеграция на федерациия и клубна статистика.
* **certification\_document** и **coach\_certification –** Тип **varchar(255)**. Тези колони

съхраняват пътищата към документите за верификация на клуб и треньор.

* **status —** тип **enum(„pending“, „active“, „rejected“)**. Тази колона съхранява статуса на

клуба.

* **verification\_status** — Тип **enum(„pending“, „verified“, „rejected“)**. Тази колона

съхранява статуса на верификация на клуба.

* **verification\_documents —** Тип **longtext(JSON)** поле, съдържащо информация за документите за верификация.
* **verification\_comments** — Тип **text**, съхраняващ коментари относно процеса на верификация.

Таблицата **club\_athletes** служи като връзка между клубовете и техните спортисти. Тя показва кои потребители (от тип "athlete") принадлежат към кои клубове, датата на присъединяване и текущия статус на членството (активно, неактивно или в процес на одобрение).

Таблица **club\_athletes:**

* **id** - тип **int(11)**. Тази колона служи като уникален идентификатор за всяка връзка клуб-спортист. Това е пъричен ключ, който автоматично се увеличава при създаване на нов запис.
* **club\_id** — Tип **int(11)**. Тази колона служи като релационна връзка към clubs.id, което е идентификатор на клуба (външен ключ към clubs.id).
* **athlete\_id —** Тип **int(11)**. Тази колона служи като релациоанна връзка към users.id -идентификатор на спортиста (външен ключ към users).
* **join\_date** — Тип **date**. Тази колона съхранява датата, на която спортистът се е присъединил към клуба.
* **status** — Тип **enum(„active“, „inactive“, „pending“)**. Тази колона съхранява статуса на членството (активно, неактивно, очакващо).
* **created\_at** и **updated\_at** — Тип **timestamp**. Тези колони съхрарняват времевите записи

на създаване и актуализиране на записа за принадлежност.

Таблицата **club\_invitations** съхранява поканите, изпратени от треньори към потенциални спортисти за присъединяване към техните клубове. Тук се проследява статусът на поканата (очакваща, приета, отхвърлена или изтекла) и включва възможност за прикачване на съобщение от треньора.

Таблица **club\_invitations:**

* **id** — Тип **int(11)**. Тази колона служи уникален идентификатор на всяка покана.

Това е първичен ключ, който автоматично се увеличава при създаване на нов запис.

* **club\_id** и **athlete\_id —** Тип **int(11)**. Тези колони служат като външни ключове за идентификатори на клуба и спортиста.
* **message —** Тип **text**. Тази колона отговаря за съхранението на съобщение от треньора към спортиста при получаване на покана.
* **status —** Тип **enum(„pending“, „accepted“, „rejected“, „expired“)**. Тази колона съхранява статуса на поканата (очакваща пращане, приета, отхвърлена, изтекла).
* **sent\_date —** Тип **datetime**. Тази колона съхранява дата и часа на изпращане на поканата.
* **created\_at** и **updated\_at —** Тип **timestamp.** Тези колони съхраняват времеви запис на създаване и последно актуализиране.

Таблицата **events** съдържа информация за всички спортни събития. Това включва име, описание, банер изображение, адрес, дати за начало и край на събитието, период за регистрация, тип на събитието (отборно или индивидуално) и връзка към създателя на събитието. Също така, има възможност за качване на разписание и схема с двойки.

Таблица **events:**

* **id —** Тип **int(11)**. Тази колона съхранява уникален идентификатор за всяко събитие.

Това е първичен ключ, който автоматично се увеличава при създаване на нов запис.

* **name –** Тип **varchar(255)**. Тази колона съхранява името на събитието.
* **description –** Тип **text**. Тази колона служи за описание на събитието. Тя се показва

само на подробната страница на събитието, но е основно поле при създаването на събитите. По тозин начин организаторите подават информация, която е важна за състезателите.

* **banner\_image** – Тип **varchar(255)**. Тази колона съхранява пътя към банер

изображението на събитието.

* **address** – Тип **varchar(255).** Тази колона съхранява адреса на събитието въведено от

организатора. То също така служи и за създаването на линк, който да пренасочи потребителите към google maps със зададения адрес.

* **start\_date**, **end\_date**, **registration\_start** и **registration\_end** — Тип **datetime**. Тези колони съхраняват началните и крайните дати за регистрация и провеждане на събититето. Избрано е този тип, защото позволява задаване не само на дата, а също така час.
* **event\_type —** Тип **enum(„team“, „free for all“)**. Тази колона съхранява вида на

състезанието, като определя дали само състезатели, които са членове на отбор, могат да се записват.

* **creator\_id —** Тип **int(11)**. Тази колона съхранява вуншният ключ към създателя на

събитието (външен ключ към users).

* **timetable\_file** — Тип **varchar(255)**. Тази колона съхранява адреса на файла в сървара на времевия график.
* **created\_at** и **updated\_at –** Тип **timestamp**. Тези колони отговарят за времевите записи на създаване и последно актуализиране.
* **bracket\_file** – Тип **varchar(255)**. Тази колона съхранява адреса на файла с жребия на

сървъра. С негова помощ, потребителите на сайта (включително и не автентикираните) могат да го свалят.

Таблицата **categories** дефинира различните категории в рамките на събитията. Всяка категория е свързана с конкретно събитие и има определена възрастова група, пол, описание, максимален брой участници и евентуален път до файл с двойките.

Таблица **categories:**

* **id**- Тип **int(11)**. Тази колона съхранява уникален идентификатор за всяка категория.

Това е първичен ключ, който автоматично се увеличава при създаване на нов запис.

* **event\_id —** Тип **int(11)**. Тази колона е външен ключ на събитието от което тази

категория е част (външен ключ към events.id).

* **name –** Тип varchar(255). Тази колона съхранява името на категорията, от което се

разбира много за нея, като пол, възраст, килограми и не на последно място дисциплина на съревнованието.

* **age\_group —** Тип: **enum(„under\_8“, „under\_12“, „under\_14“, „Cadet“, „Junior“, „under\_21“, „seniors“)**. Този колона съхранява възрастовият диапазон на категорията. Зададени са основните възрастови категории като enum, понеже те не се променят независимо от състезанието.
* **gender** - Тип enum(„male“, „female“). Тази колона съхранява пола на категорията. Той е зададен с enum, понеже вариантите винаги са два.
* **description** – Тип text. Това поле служи за коментар на категорията. Дава гъвкавост на

организаторите да уточняват до най-малкият детайл.

* **max\_participants** – Тип int. Тази колона съхранява записа на максималния борй

участници, които да бъде одобрен на категория.

* **draw\_file\_path –** Тип varchar(255). Тази колона съхранява адреса на файла на сървъра.
* **created\_at** и **updated\_at** – Тип timestamp. Тази колона съхранява времеви записи на

създаване и последно актуализиране.

Таблицата **individual\_registrations** проследява регистрациите на спортисти за индивидуални категории. Тя свързва спортиста със събитието и конкретната категория, включвайки дата на регистрация и статус (очакваща, одобрена или отхвърлена).

Таблица **individual\_registrations:**

* **id** - Тип **int(11)**. Тази колона съхранява уникален идентификатор за всяка регистрация на състезате в категорияе. Това е първичен ключ, който автоматично се увеличава при създаване на нов запис.
* **event\_id** - Тип **int(11)**. Тази колона е външен ключ към събитието за която категория

е направен записът (външен ключ към events.id).

* **category\_id** - Тип **int(11)**. Тази колона е външен ключ към категорията за която е

направен записът (външен ключ към categories.id).

* **athlete\_id** - Тип **int(11)**. Тази колона е външен ключ към идентификационния номер

на потребителя, който е записан (външен ключ към users.id).

* **registration\_date –** Тип date. Тази колона съхранява дата на създаване на записа на

поттребителя в категорията.

* **status —** Тип enum(„pending“, „approved“ , „rejected“). Тоази колона съхранява статуса на одобрение от организаторите на събието за преминаване то waiting list директно в одобрения списък на категорията.
* **created\_at** и **updated\_at** - Тип timestamp. Тази колона съхранява времевите записи на създаване и последна актуализация.

Връзките между таблиците са внимателно проектирани да осигурят цялостната функционалност на системата. Например, users и clubs са свързани чрез coach\_id, което показва кой потребител е треньор на клуба. club\_athletes създава много-към-много връзка между клубове и спортисти. Таблицата events е свързана с users чрез creator\_id, което показва кой потребител е създал събитието. categories е свързана с events чрез event\_id, което позволява групиране на категории в събития. individual\_registrations свързва спортисти, събития и категории, което позволява проследяване на участието на всеки спортист.

Базата данни включва също и някои по-специализирани таблици, като тези за отборни регистрации и участници в отборите, което разширява функционалността към отборни състезания.

Цялостната архитектура на базата данни е проектирана да поддържа цялостен процес на управление на турнири по бойни изкуства - от регистрацията на клубове и спортисти, през

организирането на събития и категории, до проследяване на регистрациите и резултатите.

### **CRUD заявки**

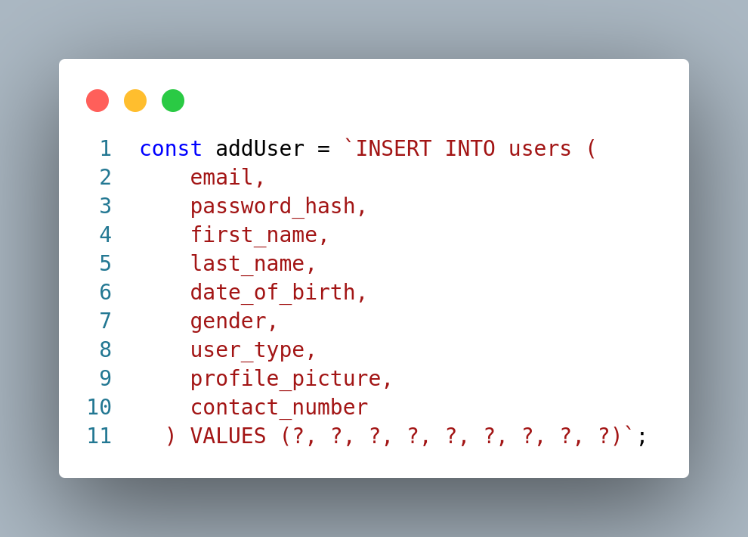
Така наречените CRUD заявки са фундаментална част от сърварната логика и работата с базата данни. Те преставляват заявки, които имат една от четири различни предназначения, а имено: **C**reate, **R**ead, **U**pdate, **D**elete. Имплементирането в приложението следва следната архитектура:

* Controller.js файлове – управляват бизнес логиката
* Queries.js файлове - съдържат SQL заявки
* Routes.js файлове - дефинират крайните точки за API
* Views.js файлове - съдържат потребителския интерфейс
* Server.js - основният файл, който настройва и стартира приложението

За всяка таблиза съществува директория със същото име в бащина директория **src**. Главната философия зад разработената архитектура е да има разделение между отделните заявки, контролери и крайни точки (endpoint, api). Това предлага на разработчиците ефективна, подредена и четлива структура, която в дългосрочен план дава лесна и сигурна основа за надграждане и осъвършенстване.

За пример даваме управлението на потребителски профил.

**Create – Създаване на потребителски акаунт:**

******

***фиг.2.2.7.1 Create заявка за създаване на потребителски акаунт***

Процесът на регистрация включва няколко ключови стъпки (виж: фиг.2.2.7.1 и фиг. 5.6):

1. Валидация на входните данни

* Системата проверява дали са попълнени задължителните полета като име, фамилия,

имейл и парола

* Извършва се проверка дали паролата и нейното потвърждение съвпадат
* Проверява се дали вече съществува потребител с този имейл

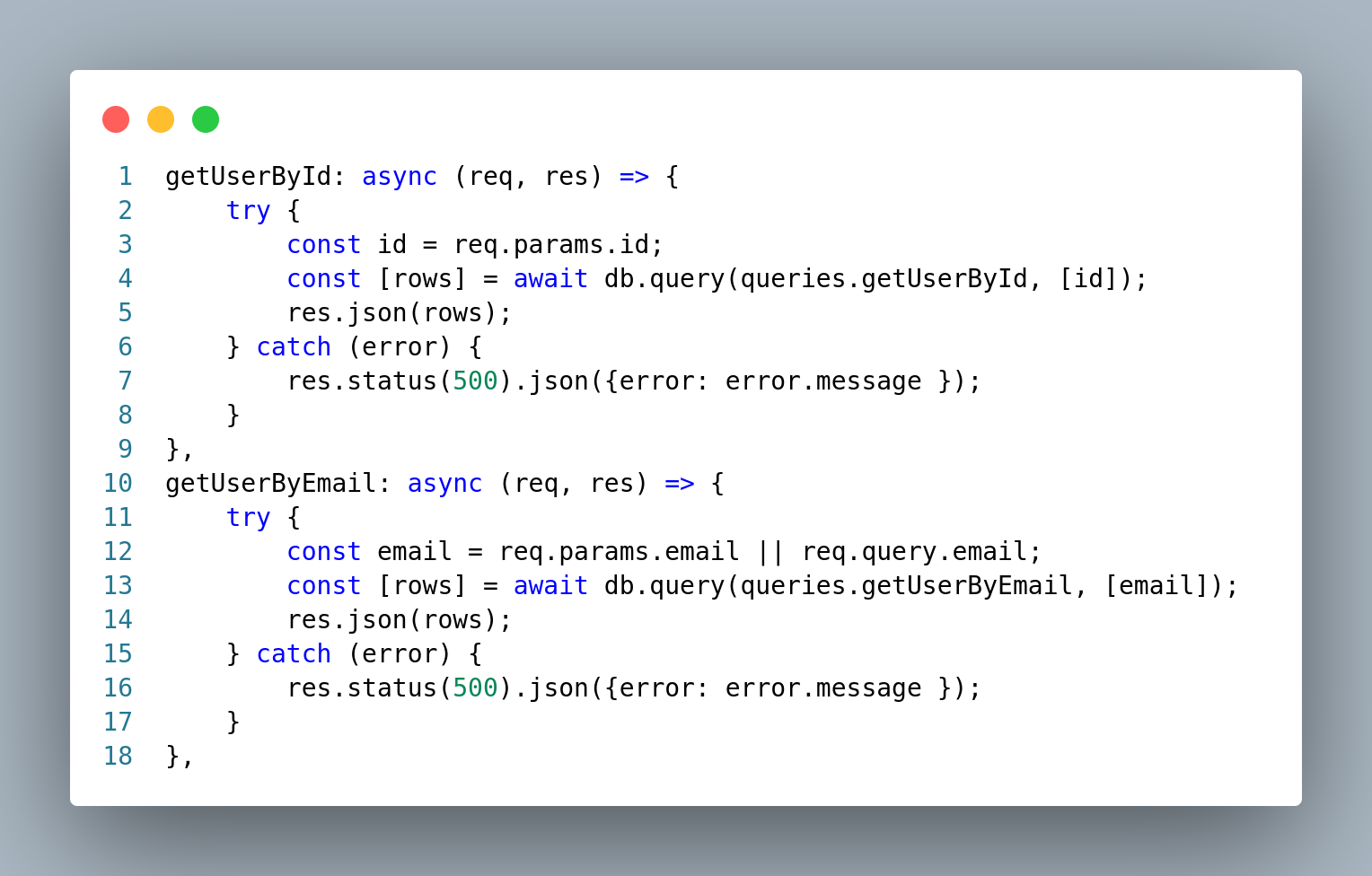
1. Обработка на входните данни

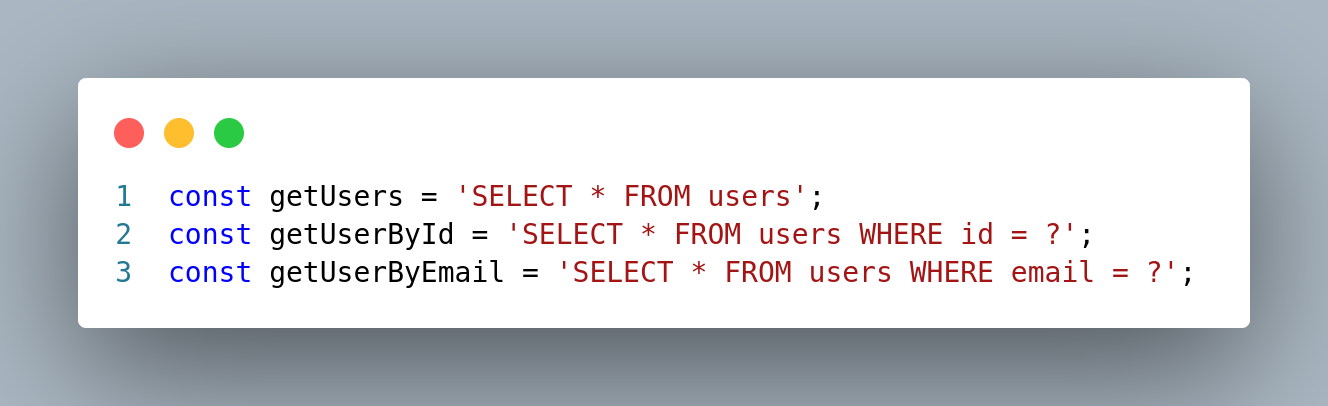
* Паролата се криптира с bcrypt хеширане за сигурност
* Ако е качена профилна снимка, тя се съхранява във файловата система
* Първоначалният тип потребител е винаги 'regular'

1. Създаване на акаунт в базата данни

* Извършва се SQL заявка за вмъкване на нов потребител
* Съхраняват се всички подадени данни, включително хеширана парола

**Read – извличане на потребителска информация от базата данни**

 ***фиг. 2.2.7.2 Controller функции, които използват тип Read sql заявки***

***фиг.2.2.7.3 Read заявки за извличане на потребителска информация***

Системата предоставя няколко начина за извличане на потребителски данни (виж: фиг. 2.2.7.2 и фиг. 2.2.7.3)

1. Извличане по ID

* Позволява получаване на пълна информация за конкретен потребител
* Използва се например при зареждане на профил

1. Извличане по имейл

* Критично за процеса на влизане
* Проверява съществуването на потребител
* Сравнява въведената парола с хеширана такава от базата

1. Извличане на всички потребители

* Административна функция
* Използва се за справки

**Update - Актуализация на потребителски aкаунт:**

Два примера за основни операции на актуализация:

1. Актуализация на профил:

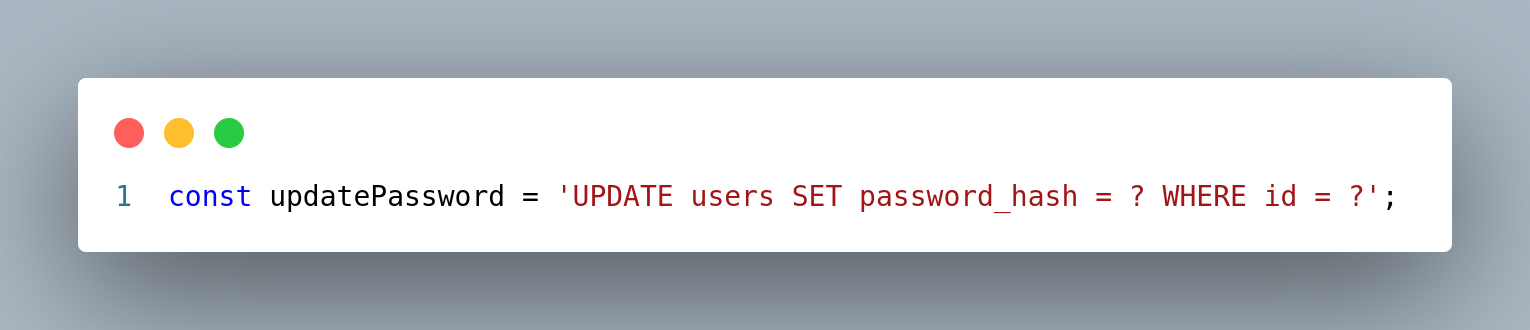


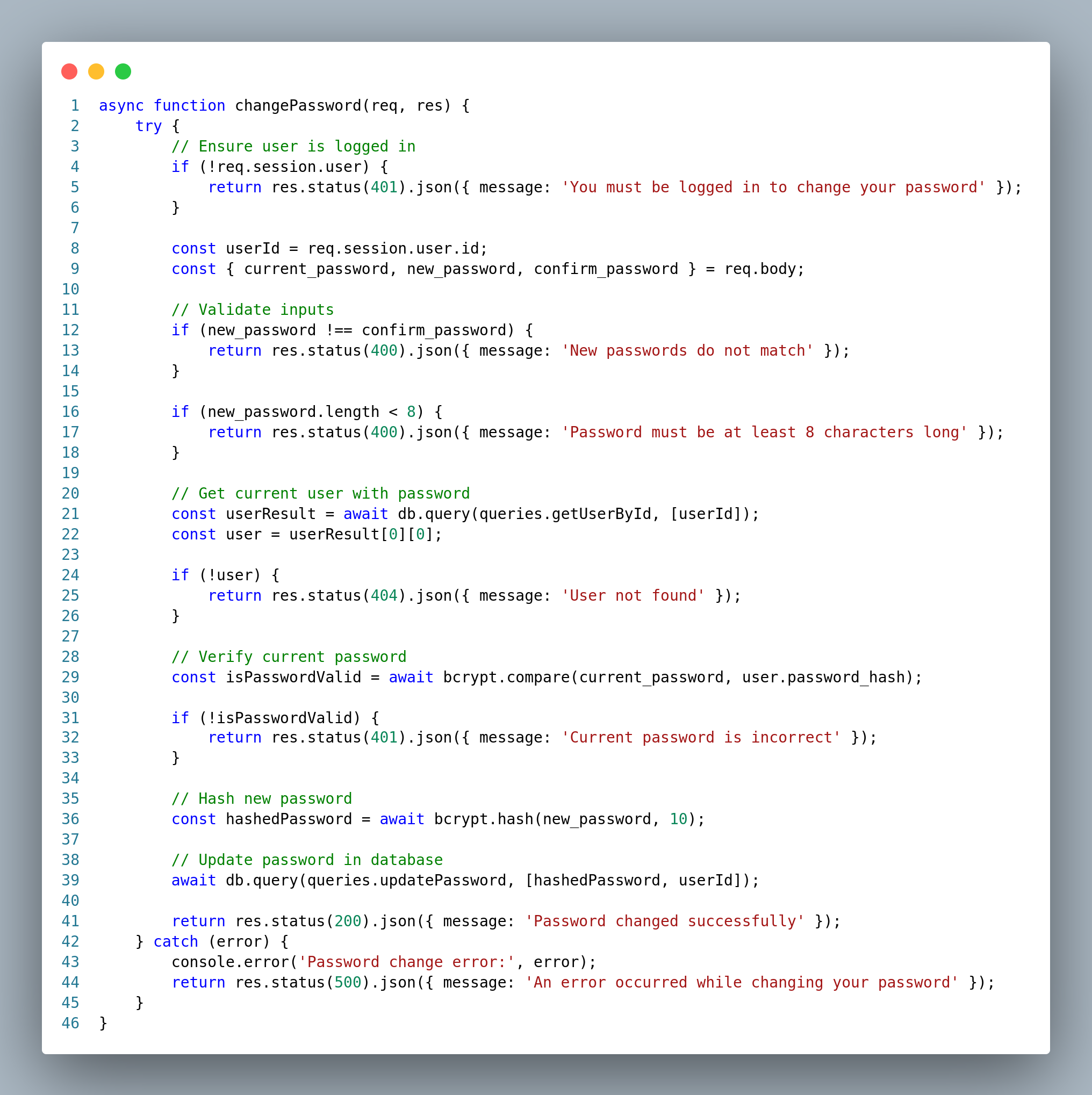
***фиг. 2.2.7.4 Update заявка отговорна за актуализирането на потребителски профил***

Актуализацията на профил позволява промяна на две имена, електронна поща,телефонаен номер и профилна снимка (виж: фиг. 2.2.7.4 и фиг. 5.5). При актуализация на профилна снимка старата се изтрива и се освобождава място, което новата да заеме. Сесията се обновява след завършване на процеса.

* + Две имена
  + Имейл адрес
  + Телефонен номер
  + Профилна снимка
* При смяна на снимка старата се изтрива
* Обновява се потребителската сесия

1. Смяна на парола:

***фиг. 2.2.7.5 Update заявка за подновяване на парола***

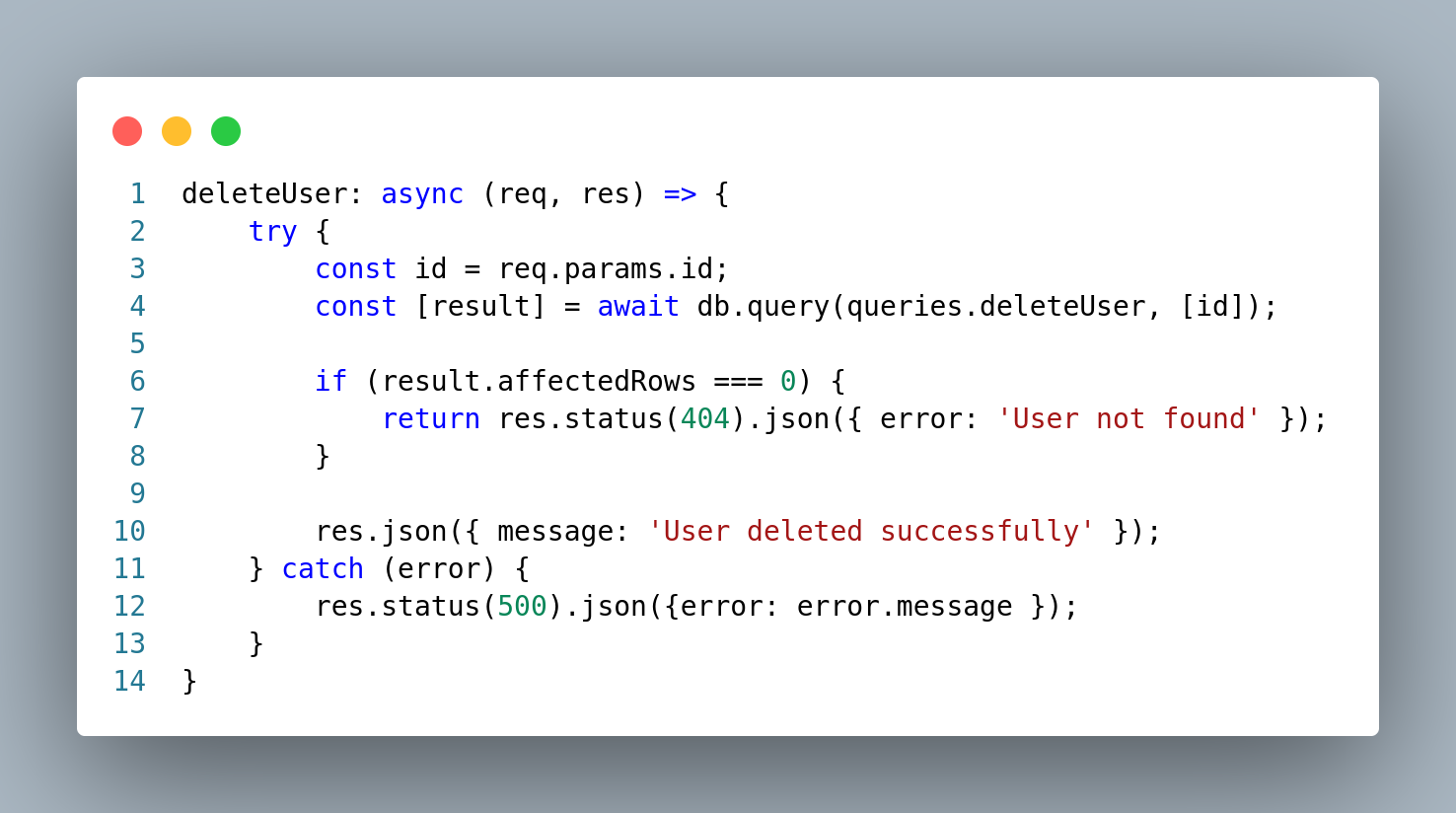


***фиг. 2.2.7.6 Controller фукция отговаряща за сменянето на паролата от потребител***

Сменянето на парола става лесно и интуитивно чрез формуляра. Стъпките, които се извърват по време на сменяне на парола са (виж: ***фиг. 2.2.7.5*** *и* ***фиг. 2.2.7.6***). Първо се изисква настоящата парола, след което трябва да преминем валидация на новата парола. Накрая паролата се хешира и се подменя старата. Нуждата от старата парола е за да предотвратим неоторизирана смяна на парола без съгласието на сбственика.

* Изисква въвеждане на текуща парола
* Извършва се валидация на новата парола
* Новата парола се хешира преди записване
* Предотвратява неоторизирана смяна на парола

**Delete - Изтриване на потребителски акаунт:**

 ***фиг. 2.2.7.7 Controller функция отговорна за изтриване на потребителски профил***

***фиг. 2.2.7.8 Delete заявка за изтриване на потребителски профил***

На фиг. 2.2.7.7 и фиг. 2.2.7.8 виждаме функция на потребителя да изтрие собствения си профил. Чрез тази заявка напълно се премахва записа на потребителския от базата данни, като се трие по уникален идентификатор, за да не се сбъркат имената и да се изтрие грешен профил.

От петте примера показани до сега виждаме в действие как работи сървърната архитектура. Бизнес логиката използва променливи вместо статически дефинирана заявка. Това прави кода по-лесно четлив, както и позволява ползването на една и съща зявка няколко пъти много лесно.

## **Тестване**

Тестването на уеб приложения е ключов етап в разработката на софтуер, който гарантира качеството и надеждността на крайния продукт. Чрез систематично тестване се идентфицират и отстраняват потенциални проблеми преди те да достигнат до крайните потребители. Нека разгледаме основните методи чрез които беше тествано приложението.

Тестване тип "Черна кутия" тестващият няма информация за вътрешната структура на приложението. Фокусът е върху функционалността от гледна точка на потребителя. Тестовете се базират на изискванията и спецификациите, като не се изисква познаване на програмния код. При този метод се проверява дали изходът отговаря на очакванията при определен вход, като се симулира реалното поведение на потребителите.

Техниките за бързо тестване включват граничен анализ, при който се проверяват граничните стойности. Друга техника е еквивалентното разделяне, което включва групиране на подобни входни данни. Тестването на потребителски сценарии също е ефективен подход. Не на последно място, проучвателното тестване позволява изследване на приложението без предварителен план.

При тестването тип "бяла кутия" тестващият има достъп до вътрешната структура и програмния код на приложението. Този тип тестване изисква познаване на програмния код и се фокусира върху вътрешната логика и структура. То позволява оптимизация на кода и откриване на скрити грешки, като е особено подходящо за откриване на проблеми със сигурността.

Техниките за бързо тестване в този случай включват тестване на изпълнението на кода, известно още като code coverage. Тестването на условията също е важен елемент от този подход. Тестването на пътя позволява проследяване на различните пътища в кода. Анализът на статичния код помага за откриване на потенциални проблеми преди изпълнение на програмата.

В резултат на тези два основни протокола за тестване качеството на продукта бе на уебприложението. Благодарение на малки и периодични тестове тип „бяла кутия“ голяма част от изпрваностите по отношение на функционалността на бизнес логиката, а с тип „черна кутия“ успешно бе вързана функционалността към графичния дизайн.

# Заключение

## **Постигнати цели**

Разработването и тестването на уеб базирано приложение за организиране на спортни събития е успешно осъществено. Проектът покрива всички предварително заложени в заданието цели, като представя елегантно софтуерно решение, което адресира основните предизвикателства при организирането на спортни събития, като създава цялостна състема за прявко взаимодействие между организатор и участник. Приложението демоснтрира и съчетава функционалност, удобство за потребителя и техничекса надеждност, като същевременно запазва минималистичен подгод към дизайна и интерфайса за максимална достъпност.   
 В процеса на разработка бяха приложени съвременни методологии и технологични решения, които осигуряват стабилност, сигурност и бързодействие на системата. Особено внимание беше отделено на потребителското изживяване, като интерфейсът беше конструиран с цел интуитивност и лесна навигация, дори за потребители без предварителен опит с подобни системи. Същевременно, техническата имплементация следва утвърдени архитектурни модели и добри практики, които гарантират стабилност и възможност за бъдещо разширяване.

Реализирани бяха следните конкретни задачи:

* Анализ на технологии за създаване на уеб базирани приложения —

извършен беше обстоен преглед на съвременните технологии за уеб разработка, в резултат на който беше избрана оптимална комбинация от HTML, CSS и JavaScript за клиентската част, Node.js и Express.js за сървърната част, и MariaDB като система за управление на бази данни.

* Възможност за създаване на профили с различни права - успешно беше

внедрена система за регистрация и управление на потребителски профили с три основни роли (обикновен потребител, треньор и състезател), всяка със специфични права и възможности.

* Възможност за създавнае на публични събития - разработена беше пълноценна

функционалност за създаване, редактиране и управление на спортни събития с различни параметри, категории и времеви периоди.

* Възможност за промяна на публични събития - имплементирана беше

възможност за модификация на съществуващи публични събития, която позволява на организаторите да актуализират информацията за събития, добавят или премахват категории, качват нови версии на жребии и графици, и управляват списъците с участници. Системата включва контролни механизми, гарантиращи, че само създателят на събитието има достъп до функциите за редактиране.

* Възможност за запазване на участие в събитията - имплементирана беше

надеждна система за записване и одобрение на участници в събития, с интуитивен интерфейс и визуална индикация на статуса на категориите.

* Възможсност за водене на хронология на събития - имплементирана беше

функционалност, позволяваща на потребителите да посещават и преглеждат събития до една година след тяхното приключване с помощта на интегрираната в приложението търсачка. Тази система осигурява достъп до архивна информация, документация и резултати от проведени състезания, което е ценен ресурс за анализ и справки.

* Технологии: MariaDB, HTML, CSS, ExpressJS, JS.

Проектът беше разработен с фокус върху добрите практики в софтуерното инженерство, включително прилагане на MVC архитектура, ясно разделение на отговорностите, внимателно моделиране на базата данни и имплементиране на необходимите мерки за сигурност.

По време на разработката се наложи преодоляването на редица технически предизвикателства, свързани с оптимизацията на зареждането при голям обем данни и осигуряването на консистентност при едновременен достъп. Тези проблеми бяха успешно решени чрез прилагане на подходящи техники и внимателно тестване.

Цялостният резултат е функционално и надеждно уеб приложение, което отговаря на зададените изисквания и предоставя добра основа за бъдещо разширяване и усъвършенстване.

## **Потенциално развитие**

Разработеното уеб приложение за управление на спортни събития предоставя солидна основа за бъдещо разширяване и обогатяване на функционалностите. Въпреки че текущата версия покрива всички поставени в заданието цели, съществуват редица възможности за доразвиване, които биха повишили полезността на платформата и разширили нейния обхват.

* + **Интеграция със спортни федерации** - разработване на системи за автоматично

синхронизиране на информация с националните и международни спортни федерации, което би осигурило официално признаване на проведените събития и отчитане на резултатите в рамките на по-широки състезателни класации.

* + **Ранг система и класиране** - внедряване на функционалност за проследяване на състезателните постижения, натрупване на точки и класиране на спортистите в зависимост от резултатите им. Това би създало допълнителна мотивация за участие и би позволило по-лесно селектиране на спортисти за национални и международни прояви.
  + **Система за проследяване в реално време** - разработване на модул за

отразяване на развитието на двубоите в реално време, като се предоставя възможност за проследяване на точки, наказания и други важни събития по време на провеждащи се състезания. Това би повишило значително интереса към платформата и би разширило нейната аудитория.

* + **Офлайн функционалност** - разработване на възможност за работа с приложението

без постоянна интернет връзка чрез имплементиране на технологии като Service Workers и IndexedDB. Това би позволило на организаторите и участниците да имат достъп до ключова информация и функционалности дори при нестабилна или липсваща свързаност, особено важно за състезания, провеждани на локации с ограничено покритие.

* + **Многоезична поддръжка** - внедряване на многоезичност, което би направило

платформата достъпна за международна аудитория и би позволило организирането на международни събития.

* + **Разширена система за комуникация** - добавяне на вътрешна система

за съобщения, известия и форуми, което би подобрило комуникацията между организатори, треньори и състезатели.

С реализирането на тези допълнителни функционалности, приложението би могло да се превърне в централна платформа за организиране и управление на спортни събития не само на местно, но и на национално и международно ниво, предоставяйки значителна добавена стойност за спортната общност.

# Източници

1. Документация на MariaDB: <https://mariadb.com/kb/en/documentation/>
2. Рaзлика между MariaDB и MySQL: https://aws.amazon.com/compare/the-difference- between-mariadb-vs-mysql/
3. Документация за NodeJS: <https://nodejs.org/en/docs>; <https://nodejs.org/api/>
4. Документация за ExpressJS: <https://expressjs.com/en/starter/installing.html>

<https://expressjs.com/en/4x/api.html> ; <https://www.npmjs.com/package/bcrypt>

1. Документация за http заявки: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
2. Документация за EJS: [https](https://ejs.co/" \l "docs)://ejs.co/#docs
3. Github Express sessions: <https://github.com/expressjs/session>
4. Документация за MVC: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
5. Документация за javascript и помощни сайтове: <https://javascript.info/>

<https://www.w3schools.com/js/>

1. Документация на bootstrap 5: <https://developer.mozilla.org/en-> US/docs/Web/API/Fetch\_API/Using\_Fetch

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

<https://getbootstrap.com/docs/5.3/components/>

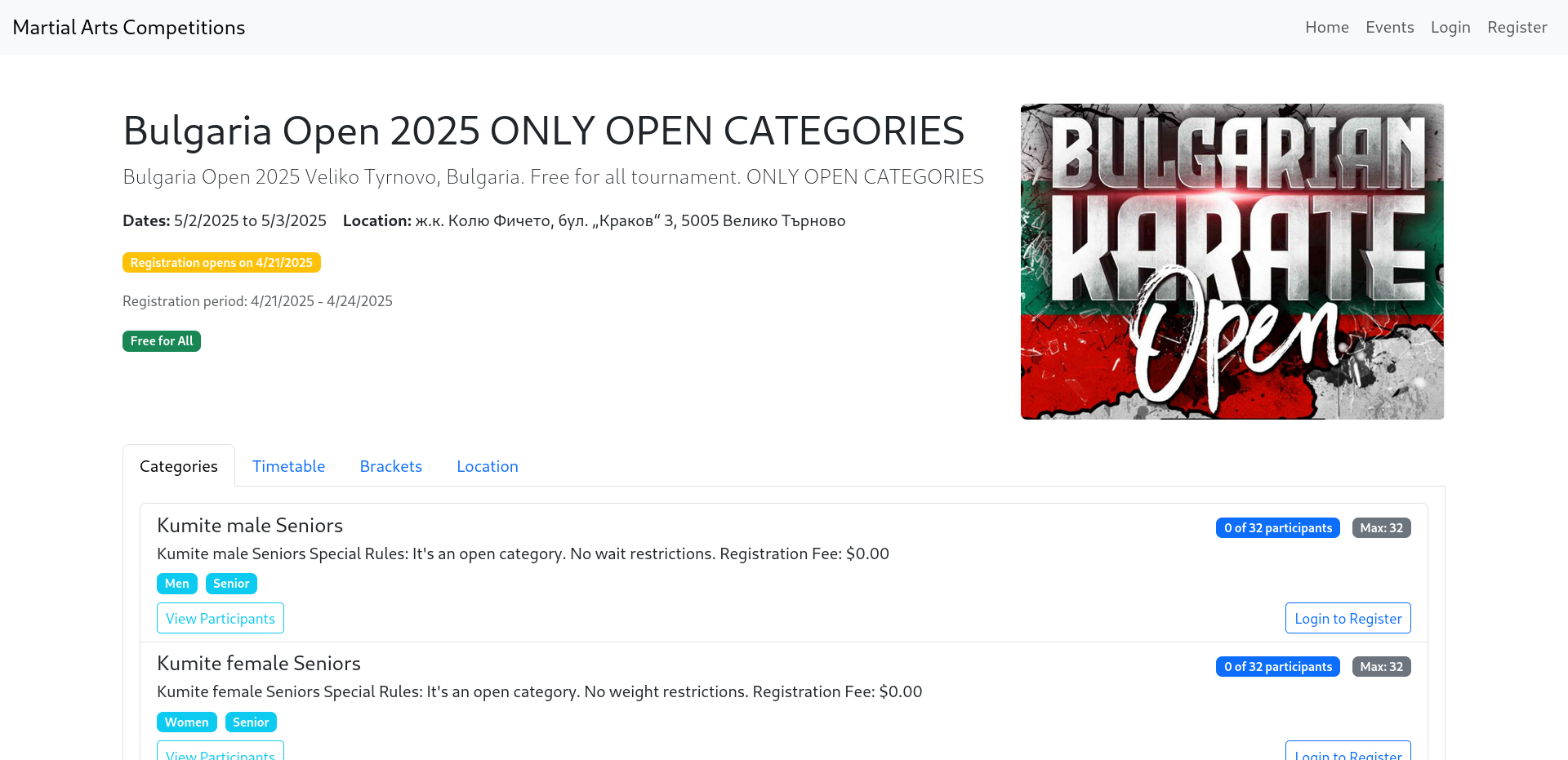
<https://getbootstrap.com/docs/5.3/utilities/>

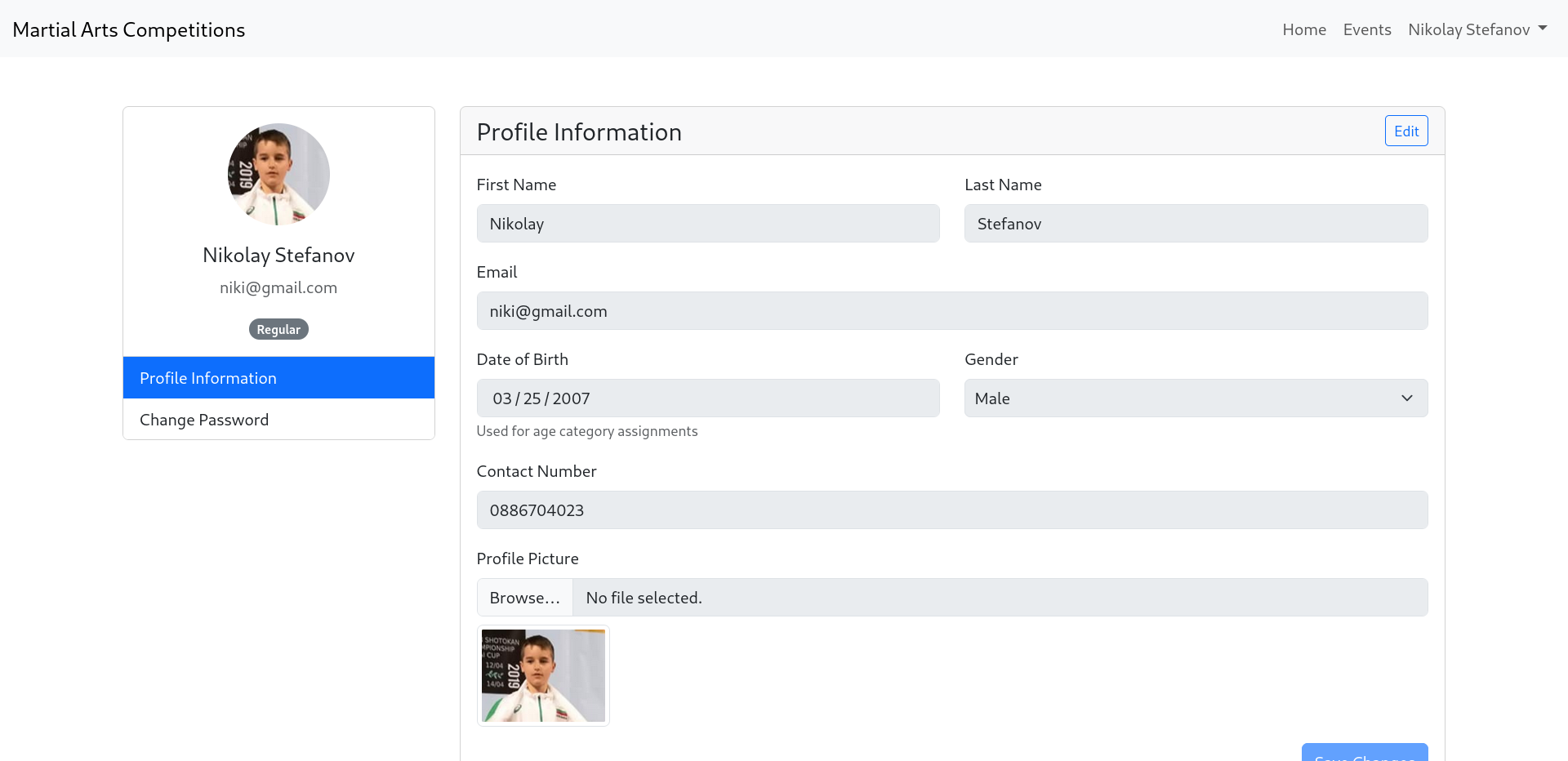
1. Документация и source code на неовим: <https://github.com/neovim/neovim>
2. Изкуственият интелект ChatGPT: <https://chat.openai.com/>

# Приложение

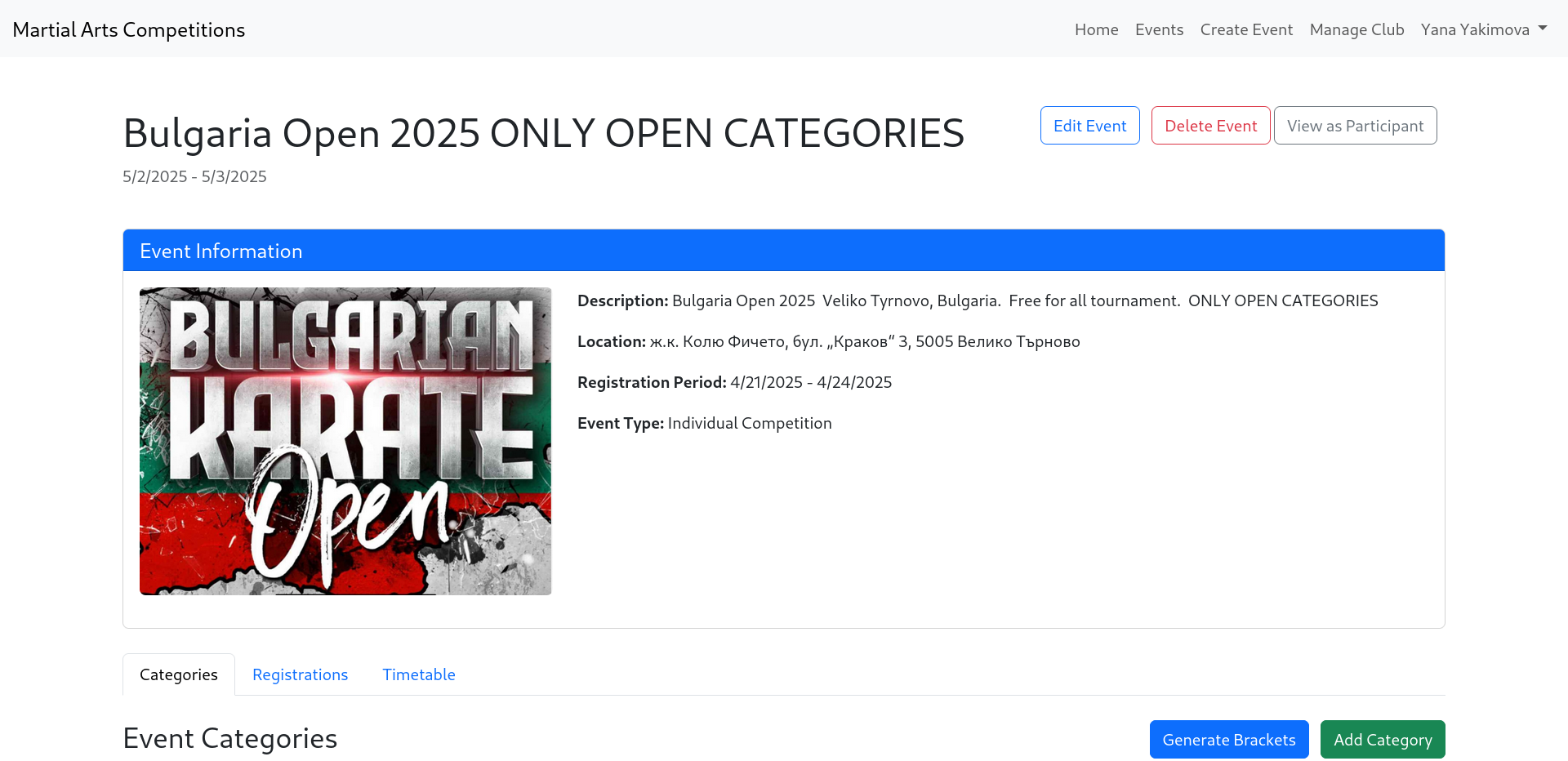
# 

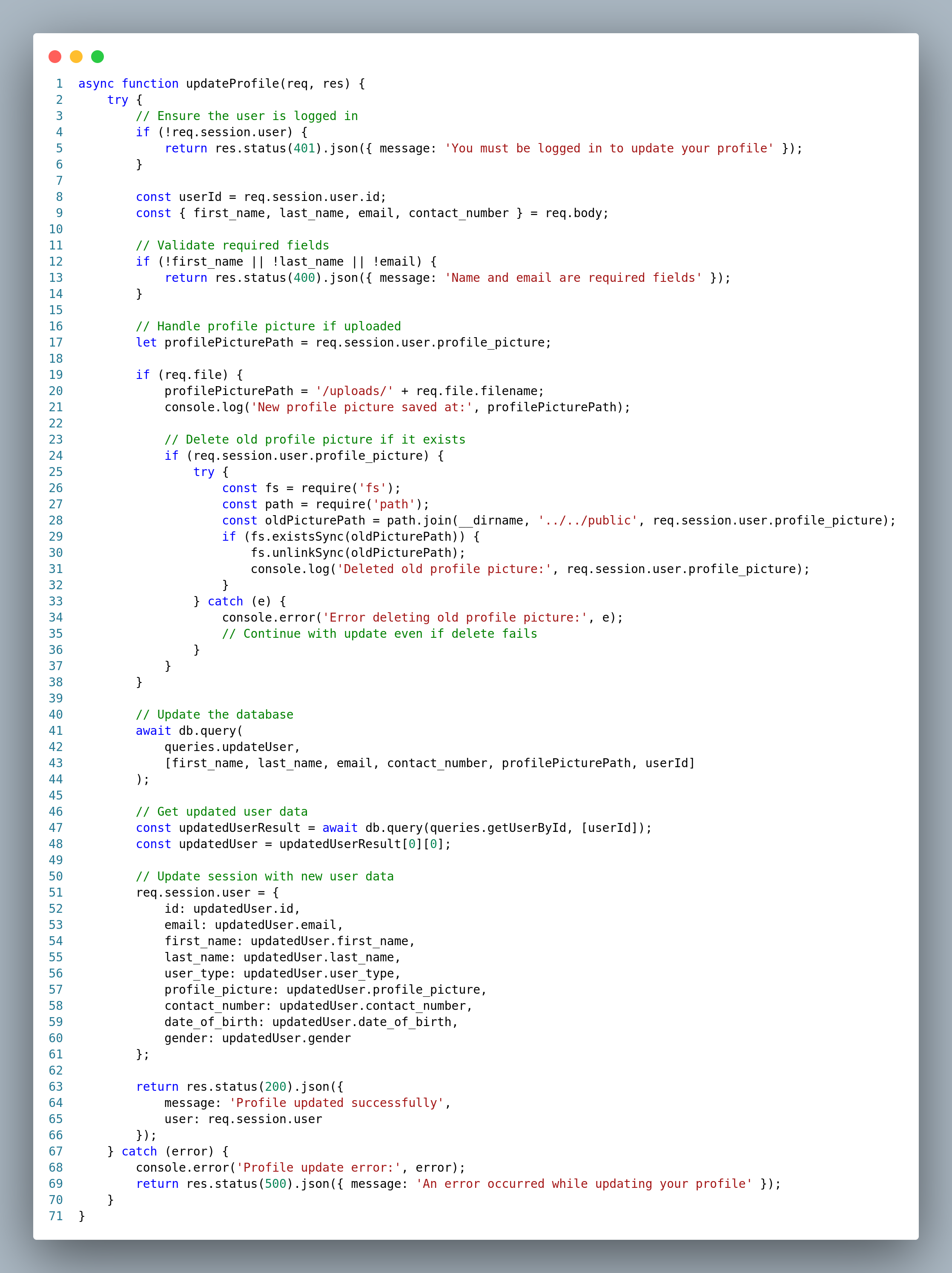
***фиг. 5.1 Страница със събития***

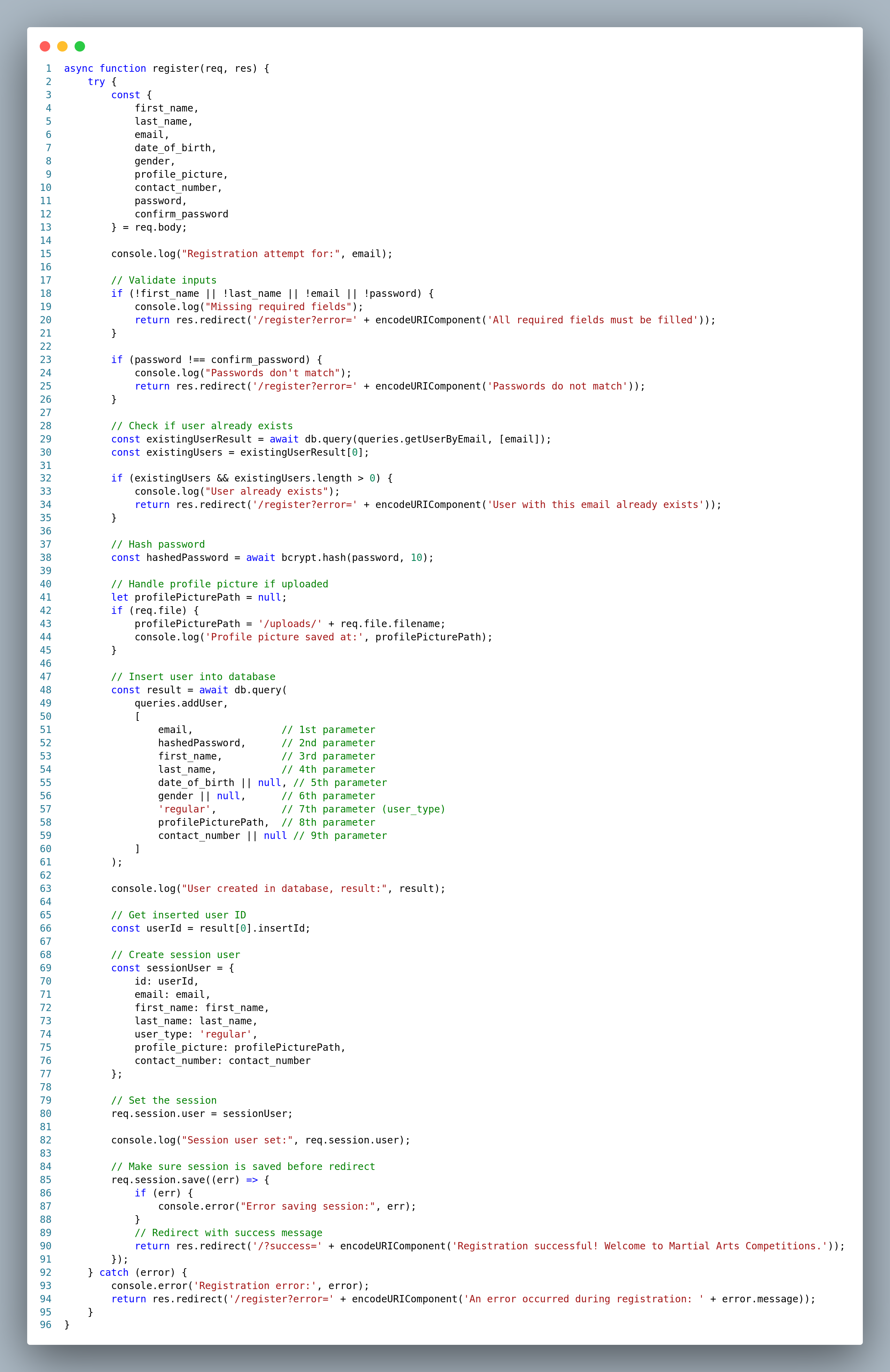
***фиг. 5.2 Страница с подробна информация за събитието***

******

***фиг. 5.3 Профилна страница***

***фиг 5.4 Част от страницата за управление на събитие***

***фиг. 5.5 Controller функциоя отговаряша за актуализирането на профил***

***фиг. 5.6 Controller фунция отговаряща за създаването на потребител***