**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ.............................................................................................................................4 1. СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ...................................................6

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ....................................................................................7

2.1 Проектирование архитектуры ПО...................................................................................7

2.2 Стек технологий................................................................................................................9

2.3 Макетирование пользовательского интерфейса...........................................................15

3. РАЗРАБОТКА АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ И СТРУКТУРЫ ДАННЫХ ..............................................................................................................................16

3.1 Разработка алгоритмов...................................................................................................16

3.2 Структура проекта..........................................................................................................19

3.3 Подробное описание модулей backend.........................................................................22

4. ТЕСТИРОВАНИЕ СИСТЕМЫ……................................................................................25 ЗАКЛЮЧЕНИЕ.....................................................................................................................30 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……....................................................31 ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

# Введение

В последнее время список сервисов и ресурсов, которые выросли в интернете небывалое множество. Интернет превратился из однообразных статических страничек в мощный инструмент интерактивности и общения с конечными пользователями. В связи с этим веб-приложения в настоящее время приобрели небывалую популярность, потому что они предлагают массу важных преимуществ, которые отсутствуют в обычных приложениях. Глядя на все происходящее в индустрии программного обеспечения можно увидеть очень большое развитие веб-приложений как отдельного звена во всей цепочке данного направления, и это не считая скорого прихода таких технологий как CSS3 и HTML5. С обычными приложениями все не так радужно, они конечно развиваются, но не так бурно, как веб-технологии. Большое множество компании переходят с обычных приложений на веб-приложения, именно потому что видят в них будущее, готовы пользоваться ими и перейти на них с обычных приложений, а за этим следует и капиталовложение, что дает мощный толчок технологиям. В связи с этим можно выделить небольшой перечень причин, по которым так происходит.

1. **Установка веб-приложений дешевле и намного проще.**

Благодаря использованию именно веб-приложений предприятия и компании могут снизить затраты на содержание ИТ отделов, которые отвечают за установку программного обеспечения и его сопровождение. В этом случае у пользователя просто-напросто лишь компьютер с браузером и соединение с интернетом или корпоративной сетью.

1. **Обновление веб-приложений дешевле и намного проще.**

Всегда большое значение имеет стоимость обслуживания ПО. Обновление ПО очень похоже на его установку, поэтому преимущества, которые были упомянуты имеют место и в данной ситуации. Для того чтобы совершить обновление веб-приложения, его необходимо обновить на только на сервере, и все сразу же смогут работать с новой версией.

1. **Веб-приложения более универсальны и практичны для конечного пользователя.**

Вам достаточно будет установить веб-приложение на сервер, работающей под любой современной ОС, и вы сможете пользоваться им через интернет на любом компьютере или мобильном устройстве, работающем под управлением MacOS, Windows, Linux или какой-либо другой ОС. Если приложения сделаны качественно то они будут работать одинаково хорошо в любом браузере, будь то Mozilla Firefox, Opera, Google Chrome, Internet Explorer или Safari.

1. **Веб-приложения облегчают организацию хранения данных.**

Если есть необходимость обращаться к одним и тем же данным из разных мест, то намного проще организовать их хранение в одном месте, вместо того чтобы разбрасывать по разным базам данных. Благодаря этому отпадает необходимость синхронизации и повысится степень их защищенности.

Именно по этим причинам веб-приложения завоевали огромную популярность, которая привела к такому уровню развития данной отрасли. Именно поэтому мы с вами сейчас все пользуемся всемирной паутиной интернет.

Основная цель работы – создание веб-приложения, предоставляющего возможности планирования спортивных мероприятий.

# АНАЛИЗ ПОСТАНОВКи ЗАДАЧИ

Программный продукт должен выполнять следующие функции:

1. Регистрация, авторизация и аутентификация пользователя
2. Организация соревнований по круговой и олимпийской системе
3. Возможность экспорта результатов (например, в формате JSON, HTML, TXT и так далее)
4. Возможность обратной связь с разработчиком

Разработать архитектуру системы на уровне подсистем и модулей. Произвести логическое проектирование базы данных, макетирование пользовательского интерфейса на основе шаблонов пользовательского опыта. Описать реализацию спроектированной системы: физическое проектирование баз данных, имплементация модулей на уровне классов и методов выбранного ЯП, экранные формы как реализация представленных ранее эскизов.

# 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## **2.1 Проектирование архитектуры ПО**

Для пользователей, которые будут пользоваться данным приложениям, будет существовать несколько уровней привилегий (по возрастанию):

* **Гость**

Имеет возможность просматривать турниры на сайте, а также имеет возможность обратной связи.

* **Пользователь**

Имеет возможности, которые имеет гость, а также возможность создавать турниры.

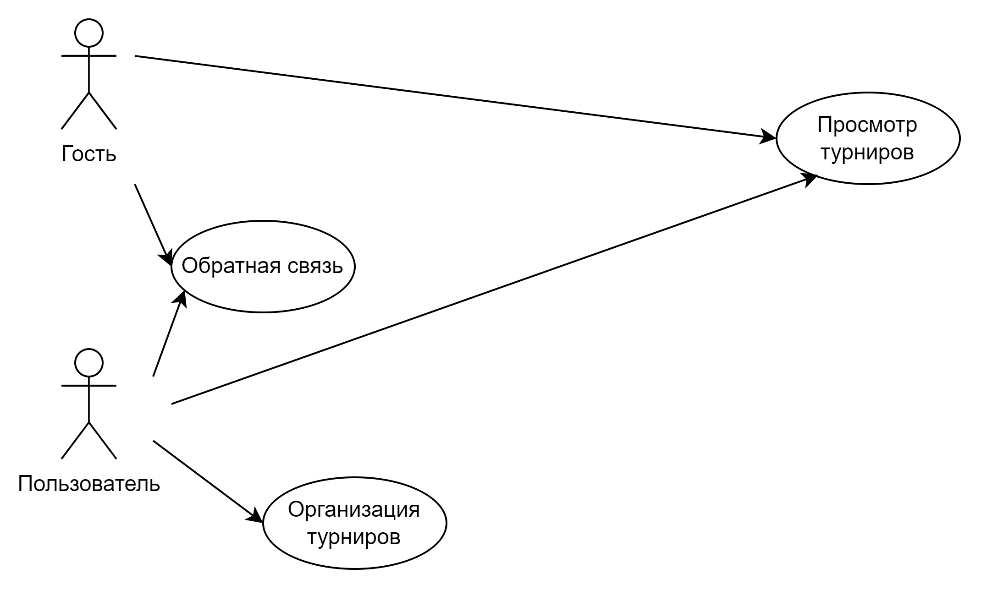


Рис. 2.1. – UML-диаграмма использования

Для хранения информации, в базе данных были созданы следующие таблицы:

* users – таблица для хранения пользователей. Имеет следующие атрибуты:
  + id – уникальный идентификатор пользователя
  + username
  + password
  + role
* tournaments – таблица для хранения турниров
  + id – уникальный идентификатор турнира
  + username\_id – идентификатор пользователя из таблицы users (является внешним ключом)
  + name – название турнира
  + description – описание турнира
  + sport – вид спорта (доступна три вида: футбол, гандбол и шахматы)
  + system – система, по которой будет проходить данный турнир (две системы: круговая и олимпийская)
* matches – таблица для хранения матчей
  + id – уникальный идентификатор матча
  + tournament\_id – идентификатор турнира из таблицы tournaments (является внешним ключом)
  + robin – количество кругов (для круговых турниров, для олимпийских значение равно 0)
  + round – номер этапа (для круговых турниров тур, для олимпийских – 1/8, 1/4 и так далее)
  + sport – вид спорта (доступна три вида: футбол, гандбол и шахматы)
  + system – система, по которой будет проходить данный турнир
* teams – таблица для хранения команд
  + id – уникальный идентификатор команды
  + tournament\_id – идентификатор турнира из таблицы tournaments (является внешним ключом)
  + team\_name – название команды
  + win – количество побед
  + draw – количество ничьих
  + lose – количество поражений
  + score – забитые мячи (для футбола и гандбола)
  + against - пропущенные мячи (для футбола и гандбола)

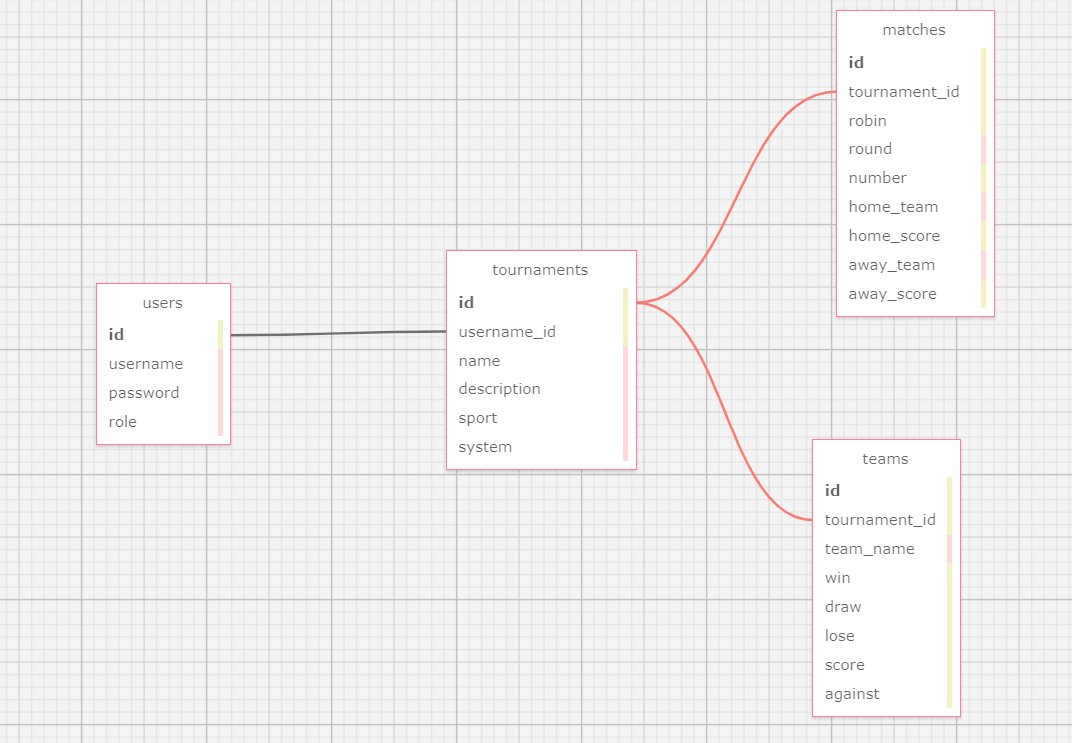


Рис 2.2 – Схема таблиц в базе данных (в таблицах не были учтены поля created\_at, updated\_at и deleted\_at)

## **2.2 Стек технологий**

Для разработки Frontend будет использоваться язык разметки HTML, язык стилей CSS, а также скриптовой язык JavaScript.

HTML – это язык разметки, который используется для отображения веб-страниц. Для создания HTML-сайта потребуется нанять профессионального разработчика, который воспользуется HTML, CSS, JavaScript и другими технологиями для построения веб-ресурса.

Весь контент HTML-сайты хранят в статичных файлах, в то время как системы управления, в том числе WordPress, используют для хранения базы данных. То есть сайты на чистом HTML-коде состоят из отдельных страниц, которые существуют реально.

**Преимущества сайтов на чистом HTML-коде**

* меньший вес;
* экономичный расход ресурсов сервера;
* не требуется обновление движка или отдельных модулей;
* практически неуязвимы к взлому;
* упрощенная система создания бэкапа;
* высокий уровень безопасности данных;
* высокая стабильность (если ошибка допущена на одной странице, она не может затронуть работу всего сайта).

CSS (англ. Cascading Style Sheets, каскадные таблицы стилей) — это простой язык дизайна, предназначенный для упрощения процесса презентации веб-страниц.

CSS обрабатывает внешний вид веб-страницы. Используя CSS, вы можете контролировать цвет текста, стиль шрифтов, расстояние между параграфами, размеры и расположение колонок, используемые фоновые изображения и цвета, макеты дизайна, варианты отображения на разных устройствах и размерах экрана. А также множество других эффектов.

Преимущества CSS:

**CSS экономит время.** Вы можете написать CSS один раз, а затем использовать одну и ту же таблицу на нескольких HTML-страницах. Вы можете определить стиль для каждого HTML-элемента и применить его ко многим веб-страницам.

**Страницы загружаются быстрее.** Если вы используете CSS, вам не нужно каждый раз писать атрибуты HTML-тегов. Просто напишите одно CSS правило для тега и примените его ко всем вхождениям этого тега. Таким образом, меньшее количество кода означает более быстрое время загрузки.

**Простота обслуживания.** Чтобы внести глобальные изменения, просто измените стиль, и все элементы на всех веб-страницах будут обновляться автоматически.

**Улучшенные стили для HTML.** CSS имеет гораздо более широкий набор атрибутов, чем HTML, поэтому вы можете сделать гораздо лучший вид своей HTML-страницы по сравнению с атрибутами HTML.

**Совместимость нескольких устройств.** Таблицы стилей позволяют оптимизировать контент для более чем одного типа устройств. Используя один и тот же HTML-документ, можно представить различные версии веб-сайта для карманных устройств, таких как PDA и сотовые телефоны, или для печати.

**Глобальные веб-стандарты.** Теперь атрибуты HTML устарели, и рекомендуется использовать CSS. Поэтому неплохо было бы начать использовать CSS во всех HTML-страницах, чтобы сделать их совместимыми с будущими браузерами.

**Оффлайн-просмотр.** Веб-приложения могут хранить CSS локально с помощью оффлайн кэша. Используя это, мы можем просматривать сайты находясь оффлайн. Кэш также обеспечивает быструю загрузку и лучшую общую производительность веб-сайта.

**Независимость от платформы.** Скрипт обеспечивает независимость от платформы и поддерживает новейшие браузеры.

JavaScript – язык программирования, являющийся прототипно-ориентированным. Он отражает язык ECMAScript, чьим прототипом изначально и являлся. Первая вариация появилась ещё в 1995 году и с тех пор постоянно совершенствовалась, пока не пришла к нынешнему виду.

Чаще этот язык используется в разработке приложений и браузерах с целью придания им интерактивности и «живости».

Преимущества JavaScript

1. Ни один современный браузер не обходится без поддержки JavaScript.
2. С использованием написанных на JavaScript плагинов и скриптов справится даже не специалист.
3. Полезные функциональные настройки.
4. Постоянно совершенствующийся язык – сейчас разрабатывается бета-вариация проекта, JavaScript2.
5. Взаимодействие с приложением может осуществляется даже через текстовые редакторы – Microsoft Office и Open Office.
6. Перспектива использования языка в процессе обучения программированию и информатике.

Для Backend-разработки буду использовать язык GoLang. GoLang — это многопоточный компилируемый язык (интерпретатор в экосистеме также предусмотрен, но на практике необходимости в нем нет: компиляция происходит мгновенно). По производительности Go почти не уступает C++ и в десятки раз превосходит скриптовые языки — такие, как JavaScript, Python, Ruby, PHP. При этом он имеет очень простой и понятный синтаксис. Код на GoLang пишется легко и приятно, чужой читается быстро и без проблем. А освоить этот язык в состоянии даже школьники.

Преимущества языка Go:

* *Высокая производительность.*
* *Простой синтаксис.*
* *Строгая статическая типизация*. У каждой переменной — свой неизменяемый тип. Если вы изначально определили ее как целое число, значит, в течение всей программы она может быть только целым числом и ничем иным. Это свойство работает на простоту кода, делает его легко читаемым, а, кроме того, сводит к минимуму риск ошибок из-за невнимательности.
* *Низкая требовательность к памяти*. В GoLang есть «сборщик мусора» — периодически те объекты, потребность в которых уже не возникнет, удаляются. Таким образом происходит автоматическая очистка памяти.
* *Большая библиотека*. В ней есть все, что необходимо для разработки. Можно также использовать библиотеки других языков — C и C++.
* *Простота параллельных вычислений*. Язык идеально подходит для создания программ, рассчитанных на многоядерные процессоры. Многопоточность, реализуемая через так называемые горутины (go routines), которые взаимодействуют посредством каналов, позволяет вызывать несколько функций практически одновременно. Эта возможность очень актуальна при создании больших и сложных программ. Средства параллельного программирования экономят время разработчика и помогают равномерно распределять ресурсы процессора.

Для реализации проекта, планируется использовать среду разработки JetBrains GoLand. Основной задачей GoLand разработчики видят предоставление такого же уровня комфорта работы и программирования на языке Go, что PyCharm обеспечивает для Python, а IntelliJ IDEA — для Java. В GoLand вас ожидают автодополнение кода, мгновенные инспекции, а также быстрые исправления и удобная навигация по коду.

Помимо основных фишек продуктов на платформе IntelliJ есть встроенные средства запуска тестов, анализ покрытия кода тестами и отладчик. Не забыли разработчики и о поддержке Git, Docker, баз данных, терминале и многих других полезных инструментов. GoLand полноценно поддерживает фронтенд-разработку и умеет работать с JavaScript, TypeScript, React, Vue.js, Angular и другими веб-технологиями.

Также я буду использовать базу данных Postgres. PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества PostgreSQL:

* поддержка БД неограниченного размера;
* мощные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
* наследование;
* легкая расширяемость.

Текущие ограничения PostgreSQL:

* Нет ограничений на максимальный размер базы данных
* Нет ограничений на количество записей в таблице
* Нет ограничений на количество индексов в таблице
* Максимальный размер таблицы — 32 Тбайт
* Максимальный размер записи — 1,6 Тбайт
* Максимальный размер поля — 1 Гбайт
* Максимум полей в записи250—1600 (в зависимости от типов полей)

Особенности PostgreSQL:

* **Функции** в PostgreSQL являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием различных языков программирования. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.
* **Триггеры** в PostgreSQL определяются как функции, инициируемые DML-операциями. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования. Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в алфавитном порядке.
* **Механизм** правил в PostgreSQL представляет собой механизм создания пользовательских обработчиков не только DML-операций, но и операции выборки. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении SQL-операции к таблице.
* **Индексы** в PostgreSQL следующих типов: B-дерево, хэш, R-дерево, GiST, GIN. При необходимости можно создавать новые типы индексов, хотя это далеко не тривиальный процесс.
* **Многоверсионность** поддерживается в PostgreSQL — возможна одновременнуя модификация БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому соблюдаются требования ACID, и практически отпадает нужда в блокировках чтения.
* **Расширение** PostgreSQL для собственных нужд возможно практически в любом аспекте. Есть возможность добавлять собственные преобразования типов, типы данных, домены (пользовательские типы с изначально наложенными ограничениями), функции (включая агрегатные), индексы, операторы (включая переопределение уже существующих) и процедурные языки.
* **Наследование** в PostgreSQL реализовано на уровне таблиц. Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут участвовать (если это не указано отдельно) в запросах к родительской таблице.

В разработке простых сайтов PostgreSQL используется несколько реже, чем MySQL / MariaDB, но всё же эта пара с заметным отрывом опережает по частоте использования остальные системы управления базами данных. При этом в разработке сложных сайтов и веб-приложений PostgreSQL опережает по использованию MySQL и MariaDB. Большинство фреймворков (например, Ruby on Rails, Yii, Symfony, Django) поддерживают использование PostgreSQL в разработке.

Что касается аппаратного обеспечения, то курсовой будет выполнен на ПК, который имеет процессор Intel Core i5-8300H CPU с частотой 2.3 ГГц, ОЗУ 8 ГБ, а также тип операционной системы: 64-разрядная OC Windows10.

## **2.3 Макетирование пользовательского интерфейса**

Интерфейс проекта будет выглядеть следующим:

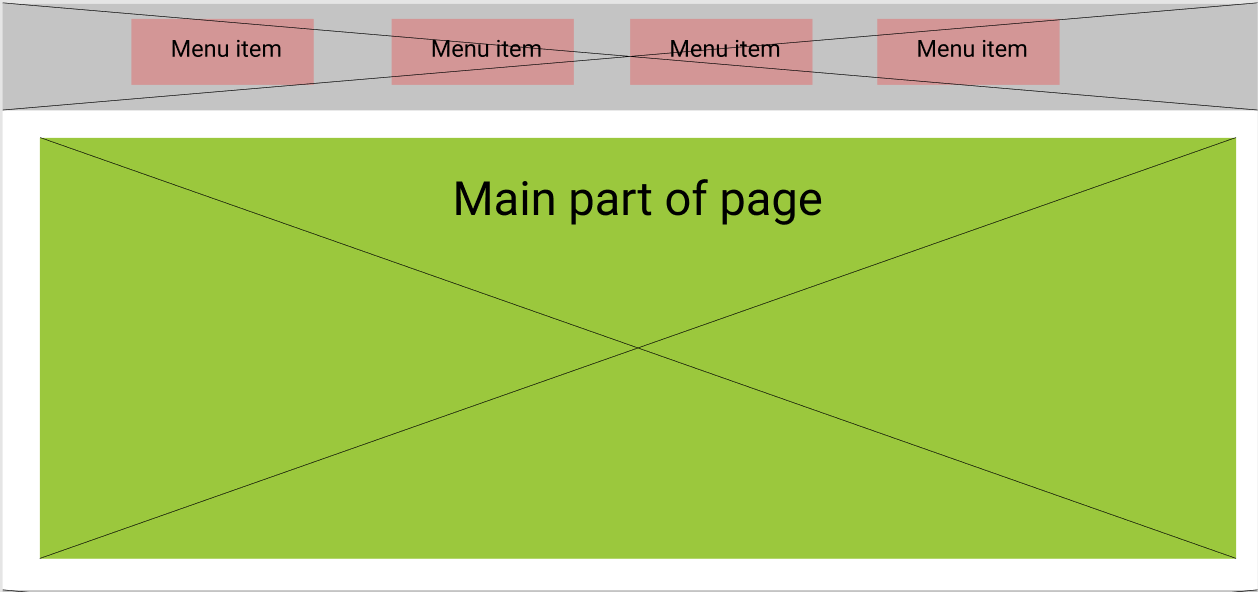


Рисунок 2.3 – Макет внешнего вида сайта

# 3 РАЗРАБОТКА АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ И СТРУКТУРЫ ДАННЫХ

**3.1 Разработка алгоритмов**

Разработаем общие алгоритмы, которые необходимы для выполнения курсовой работы.

**Алгоритм регистрации пользователя**

Исходные данные: данные, которые ввёл пользователь в форму (логин, пароль).

Алгоритм:

1. Если длина будет меньше 3 или больше 20 символов – подсвечивать ошибку.
2. Если длина пароля будет меньше 4 символов – подсвечивать ошибку.
3. Если пароль не совпадает с повторённым паролем – подсвечивать ошибку.
4. Добавление информации в базу данных с помощью SQL-запроса.
5. Переадресация на страницу авторизации.

Выходные данные: нет

**Алгоритм авторизации и аутентификации пользователя**

Исходные данные: данные, которые ввёл пользователь в форму (логин, пароль).

Алгоритм:

1. Поиск информации в базе данных с помощью SQL-запроса.
2. Если аутентификация не пройдена (логин или пароль не найдены в базе данных) – перенаправить пользователя на страницу с ошибкой. Иначе, пользователь будет авторизован.
3. Сформировать уникальный токен, который будет действовать 2 часа
4. Переадресация на главную страницу.

Выходные данные: установка в сессию данных о пользователе.

**Алгоритм изменения пароля**

Исходные данные: данные, которые ввёл пользователь в форму (старый и новый пароль).

Алгоритм:

1. Поиск информации в базе данных с помощью SQL-запроса.
2. Если старый и новый пароль совпадают, то пароль меняется, иначе ошибка
3. Переадресация на страницу профиля.

Выходные данные: нет

**Алгоритм выхода из профиля**

Исходные данные: нет.

Алгоритм:

1. Если активация кнопки «Выход из учётной записи» не была произведена в течении 6-ти часов, то выход из профиля происходит автоматически.
2. Деактивация пользователя.

Выходные данные: нет

**Алгоритм вывода всех турниров**

Исходные данные: нет.

Алгоритм:

1. Цикл перебора всех турниров.
2. Вывод турниров.

Выходные данные: список турниров.

**Алгоритм создания турнира (доступен для пользователей)**

Исходные данные: турнир, написанная редактором.

Алгоритм:

1. Ввод необходимых данных в форму.
2. Добавление информации в базу данных с помощью SQL-запроса.
3. Переадресация на станицу турниров пользователя.

Выходные данные: нет.

**Алгоритм удаления турнира (доступен для пользователей и администраторов)**

Исходные данные: турнир.

Алгоритм:

1. Поиск турнира по его id в базе данных.
2. Каскадное удаление информации из базы данных с помощью SQL-запроса.
3. Переадресация на страницу турниров данного пользователя.

Выходные данные: нет.

**Алгоритм создания расписания турнира по круговой системе**

Исходные данные: количество команд/игроков, их названия.

Алгоритм:

1. Формирование массива команд/игроков.
2. Перемешивание случайным образом.
3. Проверка на чётность. Если количество команд/игроков нечётное, то тогда в массив добавляется дополнительный элемент с именем “absent”.
4. Цикл, пока количество туров меньше количества участников, умноженное на количество кругов, минус количество участников и плюс один.
   1. Создание пар по принципу «первый-последний, второй-предпоследний» и так далее.
   2. Переход элемента на соседнюю позицию справа (за исключением нулевой позиции, которая стоит всегда на месте, и последней, которая перемещается на первую).
5. Добавление матчей в БД

Выходные данные: список матчей.

**Алгоритм создания расписания турнира по олимпийской системе**

Исходные данные: количество команд/игроков, их названия.

Алгоритм:

1. Цикл, пока количество команд не будет равен 1
   1. Формирование массива команд/игроков.
   2. Проверка на количество. Если логарифм количества команд/игроков по основании 2 – целое число, то количество матчей будет равняться количеству команд/игроков – 1, а количество раундов будет равняться логарифму количества команд/игроков по основании 2.
   3. Создание пар
   4. Уменьшение количества игроков в два раза
2. Добавление матчей в БД

Выходные данные: список матчей.

## **3.2 Структура проекта**

Структура каталогов Go (см. Рисунок 3.1):

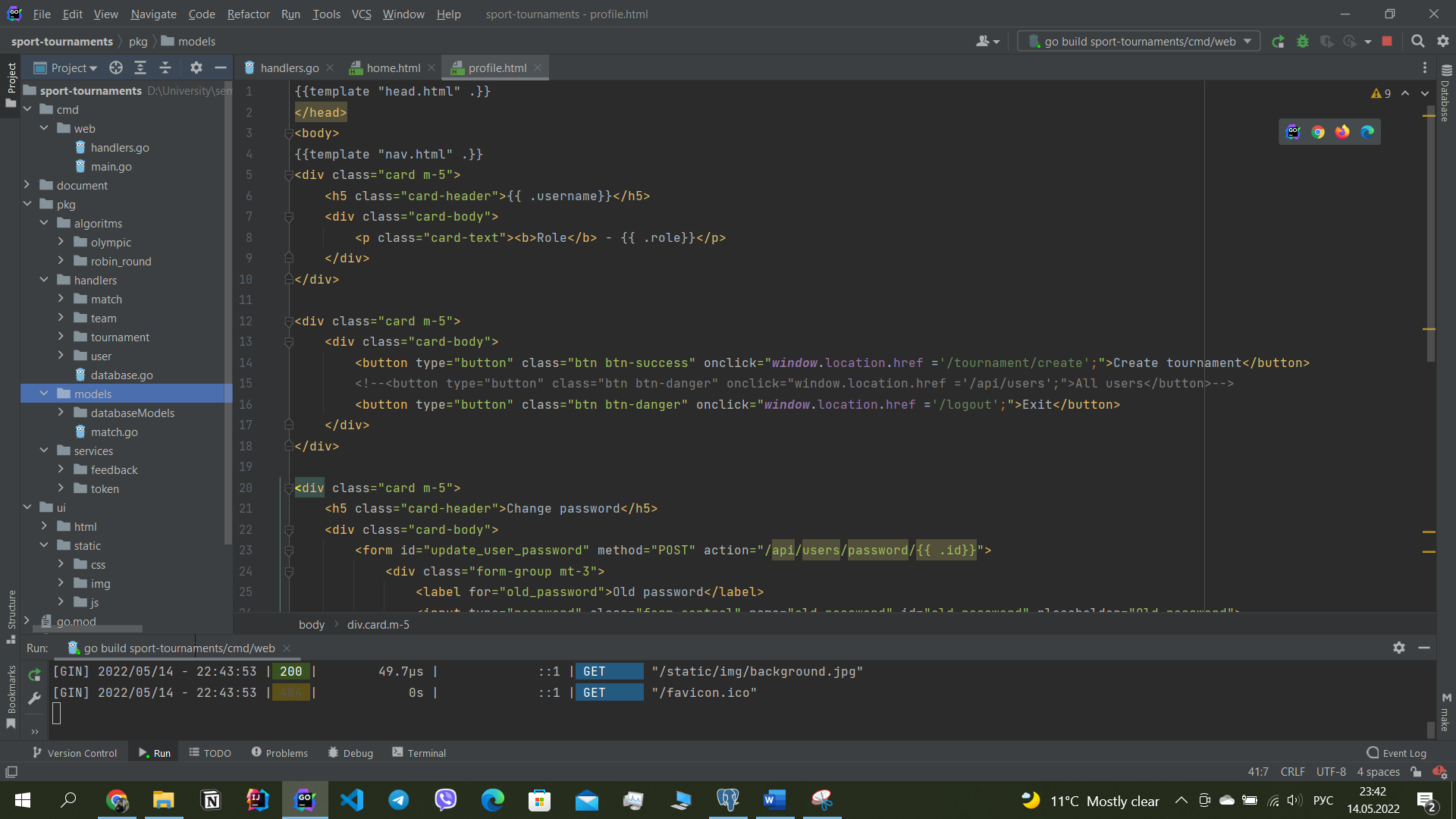


Рисунок 3.1 - структура каталогов проекта

**Домашний каталог**:

В домашнем каталоге, помимо файлов, которые созданы git, находятся файл go.mod.

Файл go.mod - это корень управления зависимостями в GoLang. Все модули, которые необходимы или будут использоваться в проекте, хранятся в файле go.mod.

Для всех пакетов, которые мы собираемся импортировать / использовать в нашем проекте, будет создана запись этих модулей в go.mod. Наличие файла go.mod экономит усилия по запуску команды go get для каждого зависимого модуля для успешного запуска проекта.

Также, в домашнем каталоге содержится 3 папки: cmd, pkg, ui.

**Каталог cmd**:

Папка cmd содержит папку web. Это означает, что у нас веб-приложение. Папка web содержит два файла – main.go и handles.go.

**Каталог pkg**:

Папка pkg содержит вспомогательный код, не зависящий от приложения в проекте. Тут содержатся папки handlers, содержащий подключение к базе данных и папки работы с таблицами, которые есть в базе данных, models, содержащий в себе структуры для работы с базой данных, algorithms, содержащий алгоритмы проведения соревнований и services, хранящий логику работы авторизации пользователя и работу обратной связи.

**Каталог ui**:

Папка ui содержит файлы HTML-шаблона для пользовательского интерфейса, используемые веб-приложением. В частности, папка ui/html будет содержать HTML-шаблоны, а папка ui/static будет содержать статические файлы (CSS, JS и изображения).

У данной структуры есть два больших преимущества:

* В структуре есть четкое разделение между Go файлами с кодом и файлами пользовательского интерфейса (HTML, CSS, JS) которые никак с Go не связаны. Весь написанный нами код на Go будет находиться исключительно в папках cmd и pkg. Корень проекта останется свободным для хранения ресурсов, не относящихся к Go. Это могут быть файлы пользовательского интерфейса, make-файлы и настройки модулей (включая наш файл go.mod). Данная структура упростит работу с веб-приложением, когда дело дойдет до развития и развертывания приложения в будущем;
* Структура отлично масштабируется, если вы хотите добавить в свой проект еще одно исполняемое приложение. Например, вы можете захотеть добавить CLI (Command Line Interface) для автоматизации некоторых административных задач в будущем. С такой структурой можно создать это CLI приложение в cmd/cli, и оно сможет импортировать и повторно использовать весь код, который была написан в папке pkg.

Модуль main.go является входной точкой в приложение.

Для определения маршрутов в приложении используется пакет Go gin-gonic/gin.

Gin это вы­со­ко­про­из­во­ди­тель­ный мик­рофрейм­ворк, ко­то­рый ис­поль­зу­ет­ся для со­зда­ния веб-при­ло­же­ний и мик­ро­сер­ви­сов. С ним очень удоб­но де­лать ком­плекс­ную кон­вей­ер­ную об­ра­бот­ку за­про­сов из мо­ду­лей - мно­го­крат­но ис­поль­зу­е­мых ку­соч­ков ко­да. Прописывается про­ме­жу­точ­ный слой при­ло­же­ния, ко­то­рый за­тем под­клю­ча­ет­ся в один или бо­лее об­ра­бот­чик за­про­сов или в груп­пу об­ра­бот­чи­ков.

Од­но из луч­ших ка­честв Go - его встро­ен­ная биб­лио­те­ка net/http, поз­во­ля­ю­щая с лёг­ко­стью со­зда­вать HTTP сер­вер.

В Go нет встро­ен­ной под­держ­ки об­ра­бот­чи­ка ро­у­тов на ба­зе ре­гу­ляр­ных вы­ра­же­ний. Вам нуж­но пи­сать код для по­лу­че­ния это­го функ­ци­о­на­ла. Од­на­ко, с ро­стом ко­ли­че­ства ва­ших при­ло­же­ний, вы бу­де­те вы­нуж­де­ны ко­пи­ро­вать один и тот же код вез­де или всё-та­ки со­зда­ди­те биб­лио­те­ку.

Для работы с базой данных используется пакет Go: gorm.io/driver/postgres.

Gorm – это реализация доступа к базе данных на языке golang. Использует библиотеку ORM (объектно-реляционное сопоставление). Благодаря этому, мы можем использовать объектно-ориентированные методы для более удобного выполнения данных в базе данных CRUD.

Функции для работы с json форматом находятся в пакете encoding/json.

Для запуска сервера и обработки запросов используется пакет Go gin-gonic/gin. Чтобы запустить сервер используется функция Run(":8080"), которая в качестве параметров принимает номер порта, на котором он будет запущен, и обработчик.

В качестве стилей приложения используется Bootstrap. Bootstrap - один из самых популярных инструментов, который используется при создании сайтов и веб-приложений.

Преимущества:

* Уменьшение количества времени, затрачиваемого на разработку
* Адаптивность
* Кросс-браузерность
* Легкость в использовании и быстрота в освоении
* Понятный код
* Единство стилей

**3.3 Подробное описание модулей backend**

**Каталог cmd**:

* main.go
  + main() – запуск программы
* handler.go
  + handler() – функция, где содержатся подфункции, хранящие обработчики событий.
  + createRouter() – функция, где подключается местоположение файлов html, css, jpg, js.
  + homePage(router) – функция, где обрабатываются события, связанные с главной страницей.
  + feedbackPage(router) – функция, где обрабатываются события, связанные с отправкой сообщения (обратная связь с разработчиком).
  + profilePage(router) – функция, где обрабатываются события, связанные с профилем пользователя.
  + userRouterGroup(router) – функция, где обрабатываются события, связанные с пользователем.
  + userTournamentGroup(router) – функция, где обрабатываются события, связанные с турниром.
  + startServer(router, port) – функция, запускающая сервер.

**Каталог pkg**:

* algorithms/olympic/olympic.go
  + Olympic(modelTeams, isMixed, idTournament) – функция, составляющая пары согласно требованиям олимпийской системы.
* algorithms/robin\_round/robin\_round.go
  + RobinRound(modelTeams, numberRobin, isMixed, idTournament) – функция, составляющая пары согласно требованиям круговой системы.
* handlers/match/createMatch.go
  + CreateSchedule(matches, numberRobin) – функция, добавляющая сформированный пары турнира в БД.
* algorithms/handlers/match/getAllMatches.go
  + GetMatchesByIdTournament(idTournament)– функция, выдающая из БД пары турнира.
* handlers/team/createTeam.go
  + CreateTeam(teamName, idTournament) – функция, добавляющая команду турнира в БД.
* handlers/tournament/createTournament.go
  + CreateTournament(nam, description, sport, usernameId, system) – функция, добавляющая турнир в БД.
* handlers/tournament/deleteTournament.go
  + DeleteTournament(idTournament) – функция, удаляющая турнир из БД.
* handlers/tournament/getAllTournaments.go
  + GetAllTournaments() – функция, получающая все турниры.
  + GetAllTournamentsByUserId(idUser) – функция, получающая все турниры определённого пользователя.
* handlers/tournament/getTournament.go
  + GetTournamentById(idTournament) – функция, получающая турнир по его идентификатору.
  + GetLastTournamentId() – функция, возвращающая последний созданный турнир.
* models/team.go
  + Team – структура для работы с таблицей в БД, хранящие данные о команде.
  + TransformedTeam – структура, для получения данных о команде из БД.
* models/tournament.go
  + Tournament – структура для работы с таблицей в БД, хранящие данные о турнире.
  + TransformedTournament – структура, для получения данных о турнире из БД.
* models/user.go
  + User – структура для работы с таблицей в БД, хранящие данные о пользователе.
  + TransformedUser – структура, для получения данных о пользователе из БД.
* models/match.go
  + Match – структура для работы с таблицей в БД, хранящие данные о матче.
  + TransformedMatch – структура, для получения данных о матче из БД.
* services/feedback/feedback.go
  + Feedback(email, text) – функция, отправляющее сообщение на почту разработчикам.
* services/token/token.go
  + GenerateToken(user) – функция, формирующая токен.
  + ParseToken(token) – функция, которая возвращает данные о пользователе, который владеет данным токеном.

# 4. ТЕСТИРОВАНИЕ СИСТЕМЫ

Данная система тестировалась на 64-разрядной операционной системе Windows 10 на ноутбуке фирмы hp с оперативной памятью 8Гб, процессором Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz.

Тестированию подлежат те задачи, которые были поставлены в разделе «АНАЛИЗ ПОСТАНОВКИ ЗАДАЧИ». В процессе тестирования была создана локальная база данных «sport\_tournaments» с следующими таблицами (см. Рисунок 4.1):

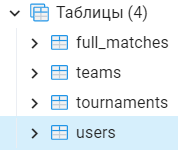


Рис. 4.1 – Список таблиц базы данных «sport\_tournaments»

**Тест 1:** «Регистрация пользователя»

Описание: проверка правильности регистрации пользователя (добавление его в БД).

Полученный результат:

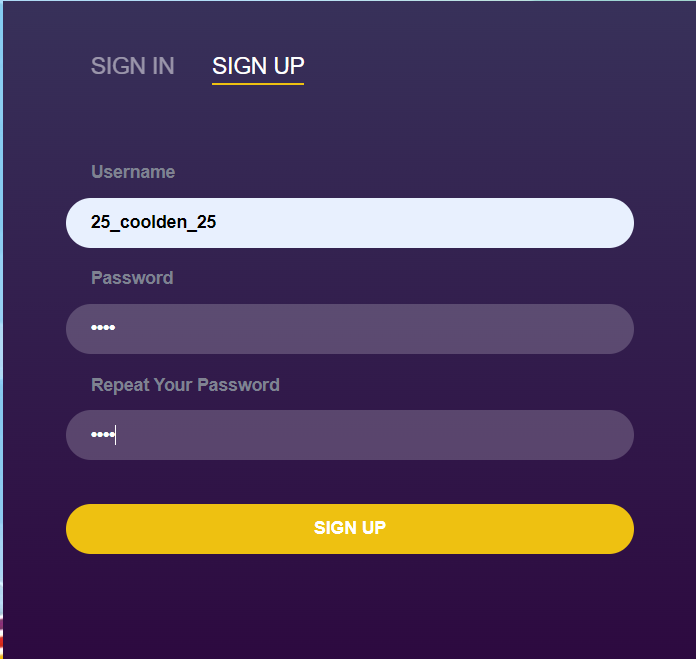


Рис 4.2 – Заполнение формы регистрации



Рис 4.3 – запись в таблице “users”

Вывод: регистрация пользователя работает корректно.

**Тест 2:** «Авторизация и аутентификация пользователя»

Описание: проверка правильности авторизации и аутентификации пользователя (проверка работы токена).

Полученный результат:



Рис 4.4 – Первоначальное меню

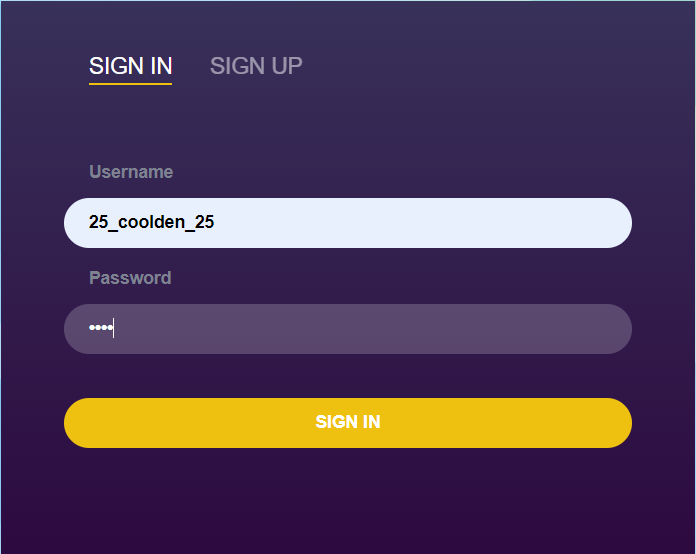


Рис 4.5 – Заполнение формы для входа



Рис. 4.6 – Новый вид меню

По истечению 2-ух часов, валидность токена была аннулирована, и меню стало такой же, как и на рисунке 4.4.

Вывод: авторизация и аутентификация пользователя работает корректно.

**Тест 3:** «Создание турнира по круговой системе»

Описание: проверка правильности создания турнира по круговой системе (добавление данных, формирование расписания).

Полученный результат:

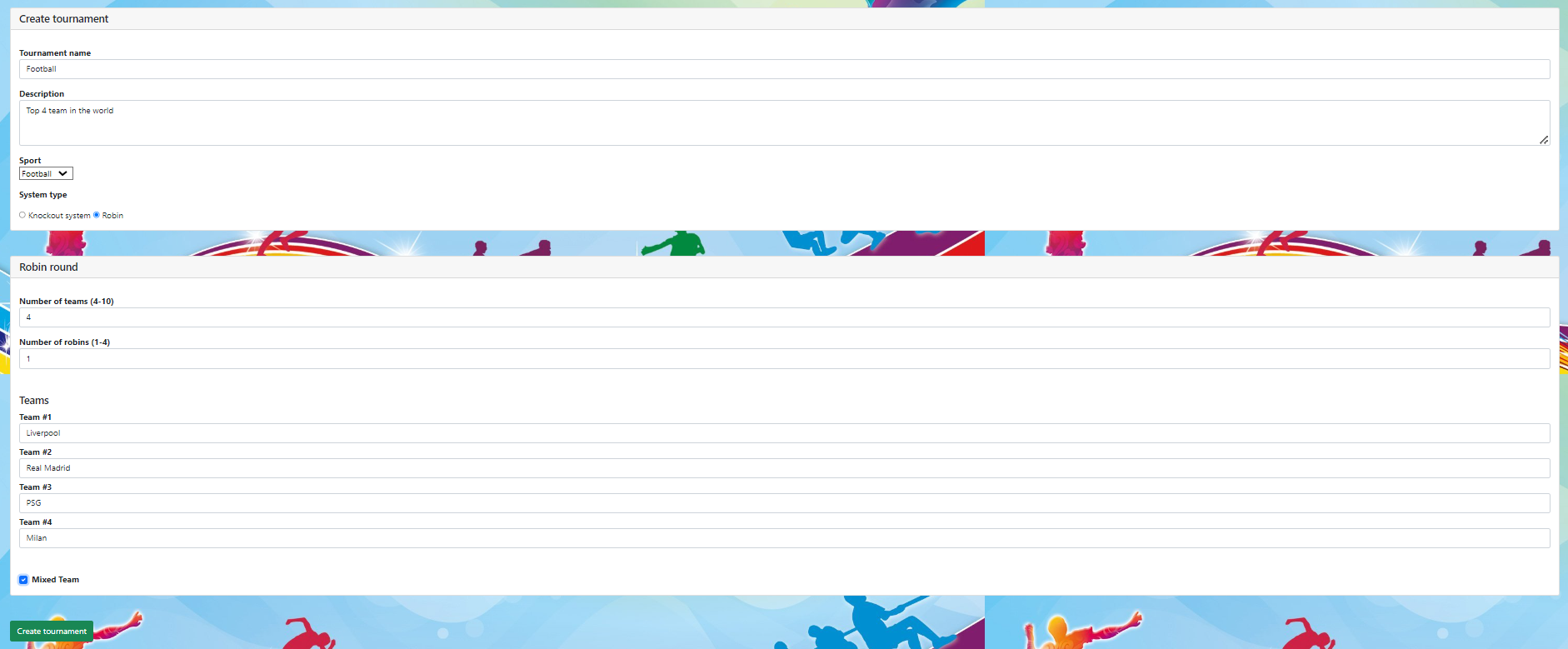


Рис 4.7 – Форма создания турнира

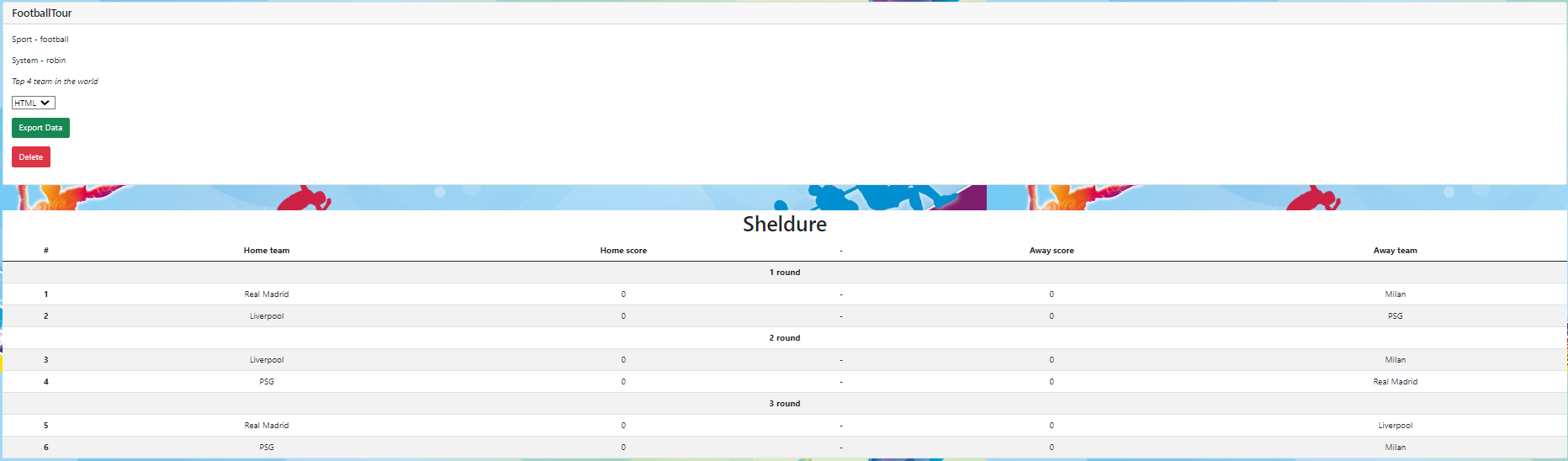


Рис 4.8 – Вывод информации о турнире

Вывод: создание турнира по круговой системе работает корректно.

**Тест 4:** «Создание турнира по олимпийской системе»

Описание: проверка правильности создания турнира по олимпийской системе (добавление данных, формирование расписания).

Полученный результат:

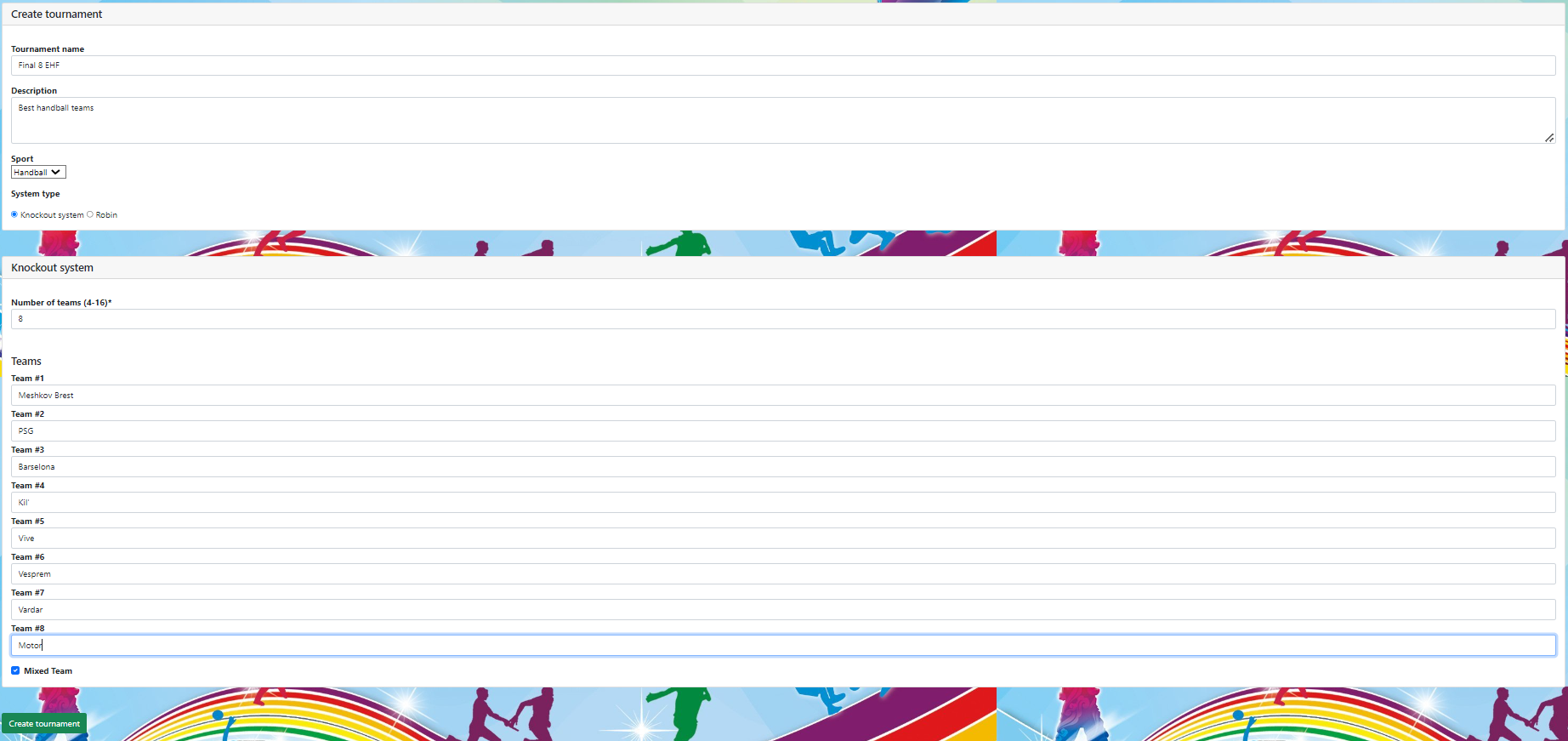


Рис 4.9 – Форма создания турнира

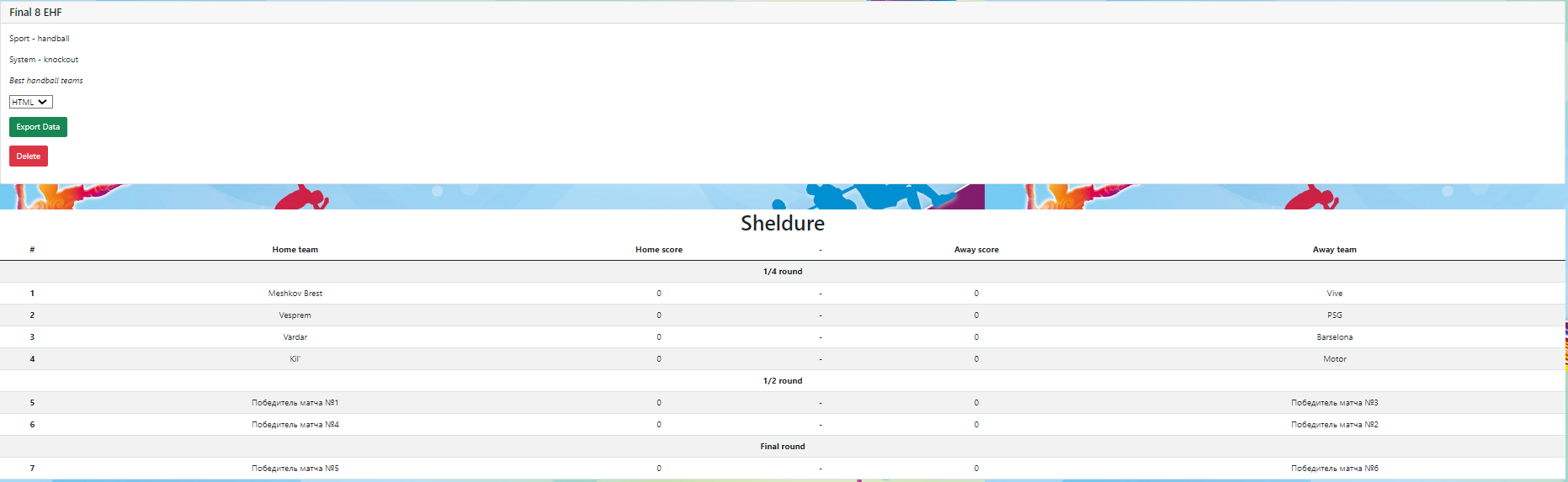


Рис 4.10 – Вывод информации о турнире

Вывод: создание турнира по олимпийской системе работает корректно.

**Тест 4:** «Проверка экспорта данных в различных данных»

Описание: проверка правильности экспорта данных в формате html, txt и json.

Полученный результат:

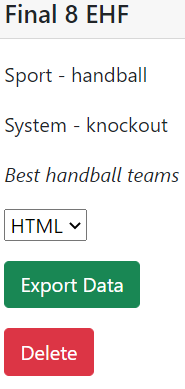


Рис 4.11 – Кнопка со списком формата

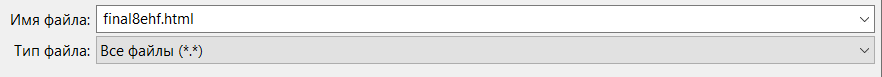


Рис 4.12 – Запуск файлового менеджера (с именем файла по умолчанию). Так делается для каждого формата

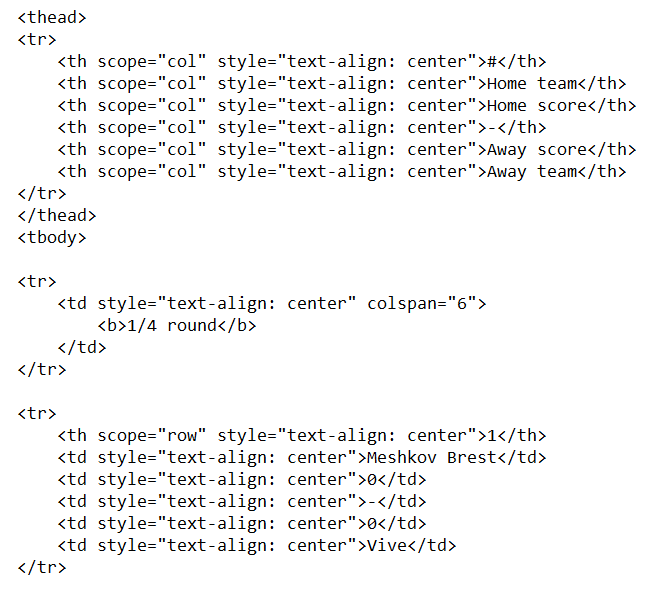


Рис 4.13 – Формат HTML

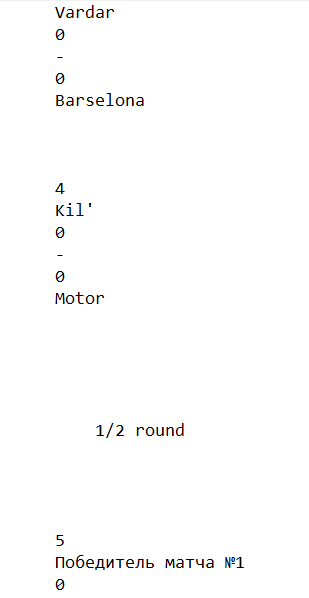


Рис 4.14 – Формат TXT



Рис 4.15 – Формат JSON

Вывод: экспорт данных о турнире работает корректно.

**Тест 5:** «Отправка сообщения на электронную почту разработчика»

Описание: проверка работы отправки сообщения на почту разработчика (обратная связь).

Полученный результат:

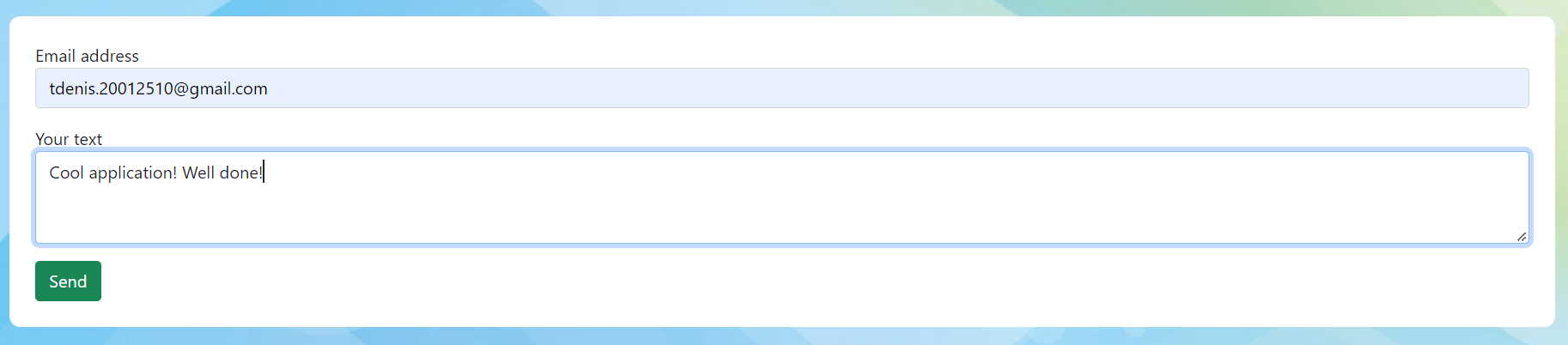


Рис 4.16 – Заполненная форма

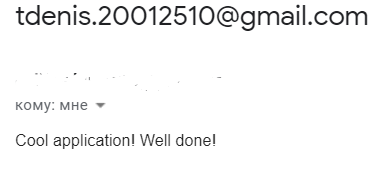


Рис 4.17 – Почта разработчика

Вывод: отправка сообщения на электронную почту разработчика корректно.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта был разработан сайт для планирования спортивных мероприятий. Система была разработана с использованием современных средств разработки.

Был получен опыт проектирования веб-приложения: выбор архитектуры, формат представления данных, способы обмена информацией, описание URI, API, проектирование базы данных. Также были приобретены навыки разработки веб-приложения на языке программирования Go и JS. Были изучены основные принципы разработки REST API сервисов.

Система соответствует требованиям технического задания. Она имеет Backend API сервер написанный на языке Go и клиентскую часть, написанную с помощью HTML, CSS и JS. Система реализована в соответствии с архитектурой «клиент-сервер», которая предполагает инициирование запроса на стороне клиента, его обработка на стороне сервера и формирование ответа, и отправка ответа обратно на сторону клиента. Данный принцип реализовывался при использовании HTTP протокола. Основными методами http-протокола являлись GET – получение данных, POST - создание данных, PATCH – частичное обновление данных и DELETE – удаление данных.

В дальнейшем на сайте планируется добавить ещё несколько дополнительных вещей, такие как:

* возможность добавление результатов матчей;
* обновление турнирной таблицы для круговой системы;
* добавление новый систем проведения турнира: смешанной и швейцарской;
* добавление новых фишек для олимпийской системы;
* улучшенный экспорт данных;
* более надёжная защита пользователей (хеширование пароля).

Данной программой могут пользоваться те, кому необходимо провести турнир, составить его расписание.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
4. ГОСТ 19.005-85. ЕСПД.

Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.

1. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.
2. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
3. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.
4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
5. ГОСТ 19.402-78. ЕСПД. Описание программы.
6. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
7. <http://sanchiz.net/blog/advantage-of-web-applications>. Преимущество веб-приложений
8. <https://vzh.ru/article/html-ili-cms-chto-luchshe-dlya-kommercheskogo-sajta> Описание преимуществ HTML.
9. <http://proglang.su/css/introduction> Описание преимуществ CSS.
10. <https://ipipe.ru/info/javascript> Описание преимуществ JavaScript.
11. <https://techrocks.ru/2021/03/29/golang-online-courses/> Описание преимуществ Go
12. <https://web-creator.ru/articles/postgresql> Описание преимуществ PostgreSQL.
13. <https://tproger.ru/news/goland-2017-3/> Описание преимуществ IDE GoLand.
14. <https://golangify.com/web-project-structure> Файловая структура веб-приложения на GoLang.
15. <https://fokusov.com/posts/razrabotka-web-prilozhenij-i-mikroservisov-na-golang-s-gin> Разработка Web-приложений и микросервисов на Go с Gin.
16. <https://russianblogs.com/article/49761651530/> Определение Gorm.