**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**имени И. Раззакова**

**Кафедра «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ»**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по теме

Веб-ориентированная система учёта занятости поля футбольного клуба Чуй

на академическую степень бакалавра

**по направлению: 710400 Программная инженерия**

**Выполнил(а): ст.гр.ПИ(б)-2-18 Эшкозиева Аида Инаятила кызы**

**Руководитель: Турсалиева Эльнура Нарынбековна**

**Бишкек 2022**

**КЫРГЫЗ РЕСПУБЛИКАСЫНЫН БИЛИМ БЕРҮҮ**

**ЖАНА ИЛИМ МИНИСТРЛИГИ**

**И.РАЗЗАКОВ атындагы КЫРГЫЗ МАМЛЕКЕТТИК ТЕХНИКАЛЫК**

**УНИВЕРСИТЕТИ**

**«КОМПЬЮТЕРДИК СИСТЕМАЛАРДЫ ПРОГРАММАЛЫК КАМСЫЗДОО»**

**кафедрасы**

**БҮТҮРҮҮЧҮ**

**КВАЛИФИКАЦИЯЛЫК ИШ**

«Чуй» футбол клубунун талаа ижарасынын эсеби веб-кадар системасы

бакалавр академиялык даражасы

**710400 «Программдык инженерия» багыты боюнча**

|  |  |
| --- | --- |
| **Аткарган** | **ПИ(б)-2-18 тобунун студенти Эшкозиева Аида Инаятила кызы** |
| **Жетекчиси** | **Турсалиева Эльнура Нарынбековна** |

**Бишкек 2022**

**Аннотация**

В данной пояснительной записке представлены результаты разработки информационной системы учета данных тренеров, администрации и участников деятельности футбольного поля и турниров.

**Объектом исследования** при написании работы послужил Футбольный клуб “FCC”- Football Club Chui.

**Целью работы** является автоматизация учета данных и взаимодействия тренеров, участников, менеджеров и администраторов Футбольного клуба Чуй.

Пояснительная записка содержит 4 главы.

Глава «Анализ требований» включает в себя результаты обзора и анализа программ с аналогичной направленностью и определение необходимых свойств разрабатываемого веб приложения.

В главе «Конструкторские работы» приведена концептуальная модель системы, а также результаты ее. Также в этой главе приведено обоснование выбора языка программирования.

В главе «Разработка документации» представлены требования к аппаратным средствам, описание программы, руководство программиста и руководство пользователя.

В главе «Экспериментальный раздел» приводится план тестирования и результаты тестирования.

В состав данной работы также входят приложения, в которых приведены описание структуры базы данных, ее таблиц, а также представлен листинг программы.

**Аннотация**

Бул түшүндүрмө катта футбол талаасынын жана турнирлердин машыктыруучуларынын, администрациясынын жана катышуучуларынын маалыматтарын эсепке алуунун маалыматтык тутумун иштеп чыгуунун жыйынтыктары келтирилген.

Чыгарманы жазууда изилдөө объектиси “FCC” футбол клубу болгон.

Иштин максаты маалыматтарды эсепке алууну жана машыктыруучулардын, катышуучулардын, менеджерлердин жана футбол клубунун администраторлорунун өз ара аракеттенүүсүн автоматташтыруу болуп саналат.

Түшүндүрмө катта 4 бөлүм камтылган.

"Талаптарды талдоо" бөлүмү окшош багыттагы программаларды карап чыгуунун жана талдоонун натыйжаларын жана иштелип жаткан веб-тиркеменин керектүү касиеттерин аныктоону камтыйт.

Бөлүм» долбоорлоо иштери системасынын түшүнүк моделин, ошондой эле анын натыйжаларын берет. Ошондой эле, бул бөлүмдө программалоо тилин тандоонун негиздемеси келтирилген.

"Документтерди иштеп чыгуу" бөлүмүндө аппараттык каражаттарга талаптар, программанын сүрөттөлүшү, программисттин колдонмосу жана колдонуучунун колдонмосу келтирилген.

«Эксперименталдык бөлүм» бөлүмүндө тестирлөө планы жана тестирлөөнүн натыйжалары келтирилген.

Бул иштин курамы, ошондой эле маалыматтар базасынын түзүмүн, анын үстөлдөрүн сүрөттөп, ошондой эле программанын листингин сунуш кылган тиркемелерди камтыйт.

Содержание

[ВВЕДЕНИЕ……………………………………………………………...………………….12](#_Toc107221227)

[Актуальность проблемы 12](#_Toc107221228)

[Цели проектирования 14](#_Toc107221229)

[ГЛАВА 1. АНАЛИЗ ТРЕБОВАНИЙ 15](#_Toc107221230)

[1.1 Обзор и анализ существующих систем 15](#_Toc107221231)

[1.2 Обоснование необходимости разработки 15](#_Toc107221232)

[1.3 Анализ требований 16](#_Toc107221233)

[1.1.1. Функциональные требования к ИС 16](#_Toc107221234)

[1.1.2. Нефункциональные требования 17](#_Toc107221235)

[ГЛАВА 2. КОНСТРУКТОРСКИЕ РАБОТЫ 18](#_Toc107221236)

[2.1 Концептуальная модель 18](#_Toc107221237)

[2.2 Разработка модели потоков данных 20](#_Toc107221238)

[2.3 Описание работы системы 22](#_Toc107221239)

[2.4 Разработка диаграммы последовательности 23](#_Toc107221240)

[2.5 Разработка диаграммы классов 25](#_Toc107221241)

[2.6 Разработка физической модели базы данных 25](#_Toc107221242)

[2.7 Разработка диаграммы компонентов 29](#_Toc107221243)

[2.8 Разработка диаграммы развертывания 29](#_Toc107221244)

[2.9 Выбор средств разработки 30](#_Toc107221245)

[ГЛАВА 3. РАЗРАБОТКА ДОКУМЕНТАЦИИ 33](#_Toc107221248)

[3.1 Требования к аппаратным и программным средствам. 33](#_Toc107221249)

[3.1.1. Требования к клиентской части. 33](#_Toc107221250)

[3.1.2. Требования к серверной части. 33](#_Toc107221251)

[3.2 Руководство программиста 34](#_Toc107221252)

[3.2.1. Средства реализации программы 34](#_Toc107221253)

[3.2.2. Входные и выходные данные 34](#_Toc107221254)

[3.2.3. Описание установки системы на серверную машину. 34](#_Toc107221255)

[3.2.4. Описание ключевых папок проекта 36](#_Toc107221256)

[3.3 Руководство пользователя 38](#_Toc107221257)

[3.3.1. Главная страница 38](#_Toc107221258)

[3.3.2. Регистрация 38](#_Toc107221259)

[3.3.3. Авторизация 39](#_Toc107221260)

[3.3.4. Личный кабинет Администратора 40](#_Toc107221261)

[3.3.5. Личный кабинет Менеджера 42](#_Toc107221262)

[3.3.6. Личный кабинет Пользователя 44](#_Toc107221263)

[3.3.7. Личный кабинет Ученика 45](#_Toc107221264)

[3.3.8. Личный кабинет Тренера 46](#_Toc107221265)

[4. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ 47](#_Toc107221266)

[4.1 План тестирования 47](#_Toc107221267)

[4.1.1. Введение 47](#_Toc107221268)

[4.1.2. Область тестирования 47](#_Toc107221269)

[4.1.3. Стратегии тестирования 47](#_Toc107221270)

[4.1.4. Начальные условия (Пререквизиты) 48](#_Toc107221271)

[4.1.5. Методы тестирования 48](#_Toc107221272)

[4.1.6. Распределение обязанностей 48](#_Toc107221273)

[4.1.7. Результаты 49](#_Toc107221274)

[4.1.8. Среда тестирования 49](#_Toc107221275)

[4.2 Сценарии тестирования 50](#_Toc107221276)

[4.3 Функциональное тестирование 50](#_Toc107221277)

[4.4 Модульное (Unit) тестирование 52](#_Toc107221278)

[4.5 Расчет метрик кода 54](#_Toc107221279)

[ЗАКЛЮЧЕНИЕ 56](#_Toc107221280)

[Список использованной литературы 57](#_Toc107221281)

[ГЛОССАРИЙ 58](#_Toc107221282)

[ПРИЛОЖЕНИЕ I 61](#_Toc107221283)

[ПРИЛОЖЕНИЕ II 63](#_Toc107221284)

# ВВЕДЕНИЕ

Футбольный клуб Чуй "FCC" – это футбольное поле в котором устраиваются турниры, игры и обучение учеников, использующие в своей работе современные IT технологии для облегчения работы персонала и пользователей полем клуба. Работают квалифицированные менеджеры и администраторы управления клубом и реализацией поля.

Официальное открытие клуба планируется к осени 2022 года, планируется основаться уже с хорошим персоналом и автоматизированной системой управления для облегчения работы персонала в городе Бишкек, касаемо расширения филиалов, на этапе реализации планов и для других городов или ПГТ в Чуйской области.

Требуется создание автоматизированной системы, использование которой будет способствовать повышению эффективности работы клуба в целом, переходу сети футбольных полей на качественно новый уровень обеспечение полем на время турниров и реализация сферы спорта по части Футбола в Чуйской области.

Актуальность проблемы

Как правило, для каждого участника все этапы записи и посещения занятий подлежат к автоматизации.

Тренер и другие сотрудники, участвующие в реализации и работе поля и клуба, вносят личную информацию, списки учеников, следят за бронями поля и распорядком проведения турниров и занятий учеников.

На ведение отчетов и составления графика занятий и проведения турниров, являющееся элементом повседневной деятельности менеджера и администратора, в некоторых случаях затрачивается более 75% времени, которое сотрудник мог бы уделить для реализации самого клуба и поля.

Часто из-за накопившейся бумажной работы, персоналу приходится брать запросы и списки команд и броней на вне рабочего времени и дополнять списки вручную на листе бумаги.

Так же в запросах на бронь и добавление команд имеется много дублируемой информации.

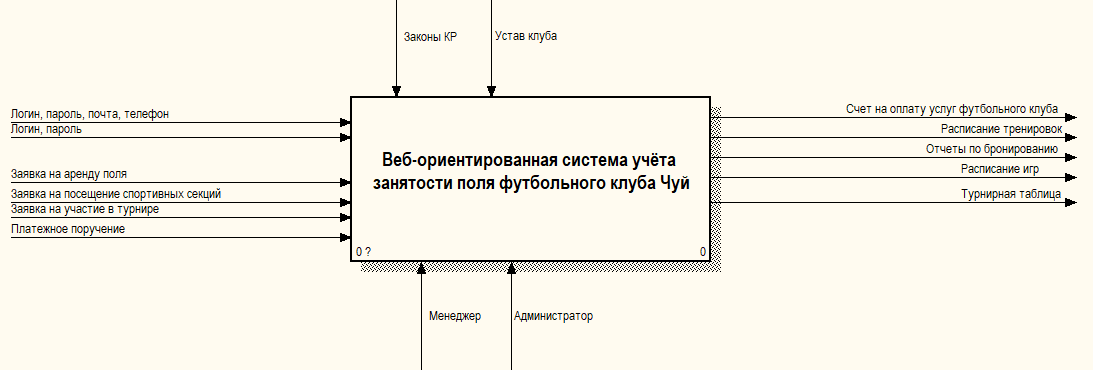
Например, те же списки команд, которые много раз дублируются хоть и информации разные, но значение их одно.

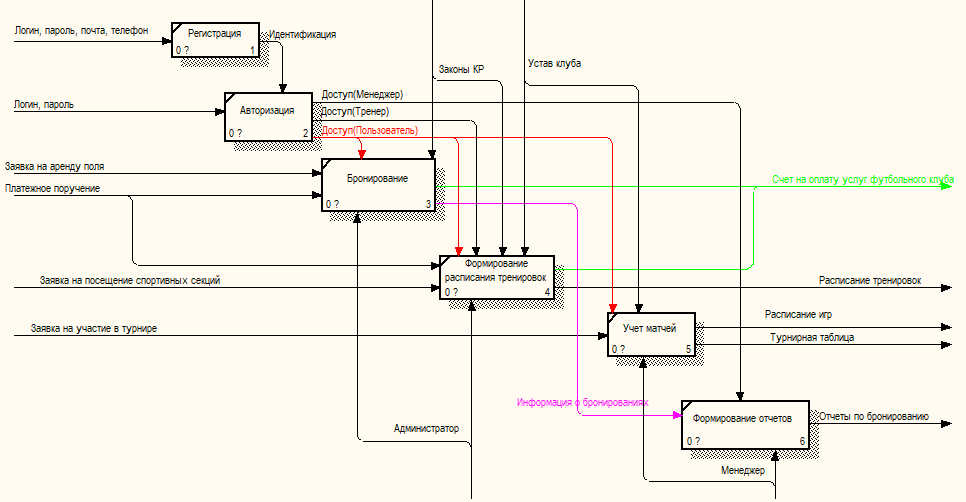
Довольно часто, на основе изучения предметной области, страдают сотрудники клуба и поля. Например, при заполнении списка турниров, сотрудники каждый раз перезаписывают результаты и списки команд вручную, и вследствие чего персонал путается и часто клиенты и тренеры просят пересмотреть время и бронь, которые запланированы на одно и тоже время.

Так же у команд возникают неудобства при получении списка команд и проверки брони у обычных игроков, часто можно наблюдать нагруженные футбольные поля с очередью или потерять листы с бронями и их графиком проведения. Оперативное сохранение графиков, броней, списков учеников, могла бы, решить данную проблемы.

Участникам лишь оставалось бы просто распечатать список команд с личного кабинета, а персоналу только добавление и отображение информации о той или иной теме в системе.

Более подробно бизнес-процессы, протекающие в клубе, представлены в виде контекстной диаграммы “To-Be” и её декомпозиций на рисунках 1.1 – 1.2

Рис.1.1. Контекстная диаграмма бизнес-процесса работы клуба.

Рис.1.2. Декомпозиция бизнес- процесса работы клуба.

Анализ бизнес- процессов показал, что в клубе при регистрации клиентов обработка большого потока информации и зачастую информация дублируется. Поэтому возникает необходимость в разработке автоматизированной системы (АС), которая позволит организовать эффективную работу клуба и футбольного поля. Благодаря использованию АС в системе “FCC” турнирная и другая информация станет доступной пользователям и участникам в режиме реального времени. Так, например, тренер может ознакомиться с графиком турниров и занятий, как только они поступили и принять решение о дальнейшем Реализации и продвижении в клубе. Кроме того, с внедрением АС исключается рутинная «бумажная» работа с потоками данных о броней, на основе которых уменьшается и время ожидания итоговых турнирных таблиц и информации о брони.

Цели проектирования

Целью ВКР является разработка автоматизированной информационной системы учета данных турниров занятий и броней, которая позволит:

* Снять с менеджера непрофильной нагрузки или существенно ее уменьшить, что позволит ему уделять больше времени непосредственно работе по реализации футбольного клуба.
* Сократить (а местами - и полностью исключить) рутинные операции по распределению графику и турнирных таблиц вручную, что позволит существенно уменьшить ошибки, повысить качество и информативность в период проведения занятий и турниров,
* Повысить эффективность информационного взаимодействия между всеми пользователями клуба и вне,

Разрабатываемая система должна решать следующие задачи:

Основная функция рассматриваемой системы состоит в бронировании футбольного поля, записи на занятия.

В программе должно быть автоматизировано бронирование поля, запись на занятия.

Онлайн-запись позволит пользователям легче забронировать футбольное поле, записаться на занятия и соответственно будет больше новых клиентов.

Система также будет составлять расписание с групповыми занятиями.

Поможет контролировать историю бронирований, получать отчеты по клиентам и бронированиям, что поможет лучше контролировать важные показатели бизнеса.

1. АНАЛИЗ ТРЕБОВАНИЙ

1.1 Обзор и анализ существующих систем

В Кыргызской республике, нет аналогичных существующих систем для подобных полей и клубов.

Но есть аналоги в других странах, например Россия.

<https://raketa-tennis.ru/booking> - Теннисный клуб "РАКЕТА“

Преимущества:

* Приятно оформленный и интуитивно понятный сайт
* Адаптивный интерфейс
* Удобно реализована аренда теннисного корта

Недостатки:

* Этот сайт только для теннисного клуба "РАКЕТА“
* Содержит только бронирования

<https://findsport.ru/> объединил спортивные секции и площадки крупнейших городов России.

Преимущества:

* Множество спортивных секций и площадок
* Множество видов спорта и занятий
* Есть карта со всеми площадками

Недостатки:

* Включает в себя только площадки городов России
* Содержит только бронирования

1.2 Обоснование необходимости разработки

Нет каких-либо систем для управления клубами и футбольными полями в Кыргызстане.

Таким образом, становится актуальной реализация системы в виде веб-приложения. Это позволит сделать ее доступной как для тренеров и менеджера, так и для обычных пользователей. Такой подход сразу откроет ряд преимуществ:

* Во-первых, такой способ оборота информации доступна пользователям из любого места без установки дополнительного программного обеспечения.
* Во-вторых, такой оборот информации позволит пользователям легко отслеживать доступное для бронирований время, процесс составления турнирных таблиц и графика занятий, а менеджерам корректировать его по мере необходимости.

Опыт, полученный в результате изучения сферы работы подобных структур, позволяет создать автоматизированную систему, которая будет отвечать современным требованиям и соответствовать ожиданиям пользователей и персонала.

1.3 Анализ требований

В ходе анализа предметной области были выявлены следующие пользователи системы:

* Пользователь
* Тренер
* Ученик
* Менеджер
* Администратор
  + 1. Функциональные требования к ИС

1. Функционал безопасности и разграничения прав доступа:
2. Регистрация пользователя
3. Авторизация пользователя с определёнными правами
4. Функционал Пользователя:
5. Бронирование поля
6. Отмена бронирования
7. Просматривать информацию о свободном времени для бронирования
8. Записаться в учебную группу
9. Зарегистрировать на турнир команду
10. Просмотр турнирной таблицы
11. Просмотр расписания игр
12. Просмотр бронирований пользователя
13. Функционал Тренера:
14. Просматривать список учеников
15. Просматривать расписание тренировок
16. Функционал Ученика:
17. Просматривать расписание тренировок
18. Просматривать список учеников своей группы
19. Функционал Менеджера:

1.Просмотр отчетов по бронированию

2.Задать условия для скидок и бонусов

3.Создать турнир

4.1.Создать матч

4.2.Принять заявку на участие

5.Добавить администратора

6.Создание групп

7.Назначение тренера группе

8.Создание расписания групп

9.Создать расписание тренировок

1. Функционал Администратора:

1.Создать бронирование

2.Отменить бронирование

3.Записать ученика в группу

* + 1. Нефункциональные требования
* Все компоненты ИС “FCC” должны функционировать в пределах единого логического пространства, обеспеченного интегрированными средствами серверов данных и серверов приложений.
* Информационный обмен между системами должен осуществляться через единое информационное пространство и посредством использования стандартизированных протоколов и форматов обмена данными.
* Реализация ИС должна быть в виде веб-приложения с возможностью доступа через интернет.

1. **КОНСТРУКТОРСКИЕ РАБОТЫ**
   1. Концептуальная модель

Для того чтобы проанализировать и специфицировать общие требования к функциональному поведению проектируемой системы была разработана концептуальная модель.

На Рис.2.1. – 2.2. представлена концептуальная модель АС в виде диаграмма вариантов использования, на которой отображены все пользователи системы, а также варианты использования АС, с которыми они связаны. На диаграмме приведены следующие варианты использования:

* *Создать бронирования поля*–доступно всем пользователям.
* *Отменить бронирования поля* –доступно всем пользователям.
* *Записаться на занятия* – доступно всем пользователям.
* *Записать ученика на занятия* –доступно только администратору.
* *Добавить администратора* –доступно только менеджеру.
* *Просмотр отчета по бронированиям* –доступно только менеджеру.
* *Создать турнир* –доступно только менеджеру.
* *Принять заявку команды*–доступно только менеджеру.
* *Создать матч* –доступно только менеджеру.
* *Просмотр списка учеников* –доступно только тренеру и ученику.
* *Просмотр расписания тренировок* –доступно только тренеру и ученику.
* *Зарегистрировать на турнир команду*–доступно только тренеру и ученику.



Рис.2.1. Диаграмма вариантов использования ролей Менеджер и Администратор.



Рис.2.2. Диаграмма вариантов использования ролей Ученик, Тренер, Пользователь.

* 1. Разработка модели потоков данных

На рис.2.3. – 2.4. представлена модель TO-BE в виде диаграммы потоков данных (Data-Flow-Diagram), описывающей внешние сущности, хранилища, процессы, потоки информации, протекающие в разрабатываемой АС “FCC”.

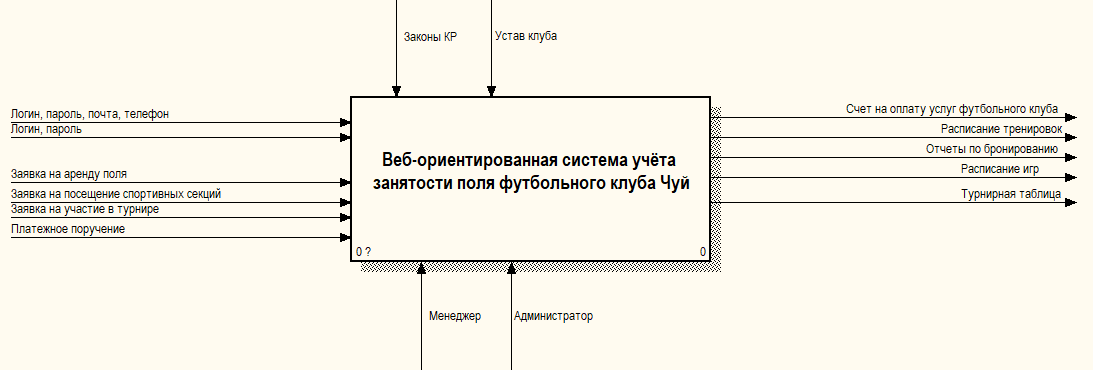


Рис.2.3. Контекстная диаграмма потоков данных в АС.

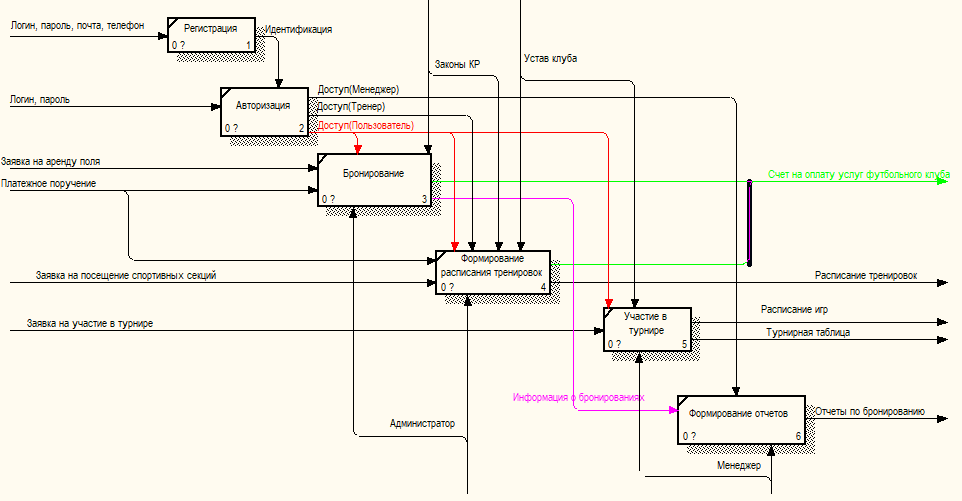


Рис.2.4. Контекстная диаграмма потоков данных в АС.

* 1. Описание работы системы

На рис.2.5. показана Activity Diagram (диаграмм деятельности), которая описывает работу системы при бронировании поля пользователем.



Рис.2.5. Бронирование поля.

На рисунке 2.6. показана Activity Diagram (диаграмм деятельности), которая описывает работу системы при создании матча Менеджером.



Рис.2.6. Создание матча.

* 1. Разработка диаграммы последовательности

Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.



Рис.2.7. Диаграмма последовательности системы учета футбольного клуба.

* 1. Разработка диаграммы классов

На рис.2.8. отображена статическая структура ИС в виде диаграммы классов. Система реализована через взаимодействие объектов из 12 классов. На диаграмме классов отображены только основные классы системы.

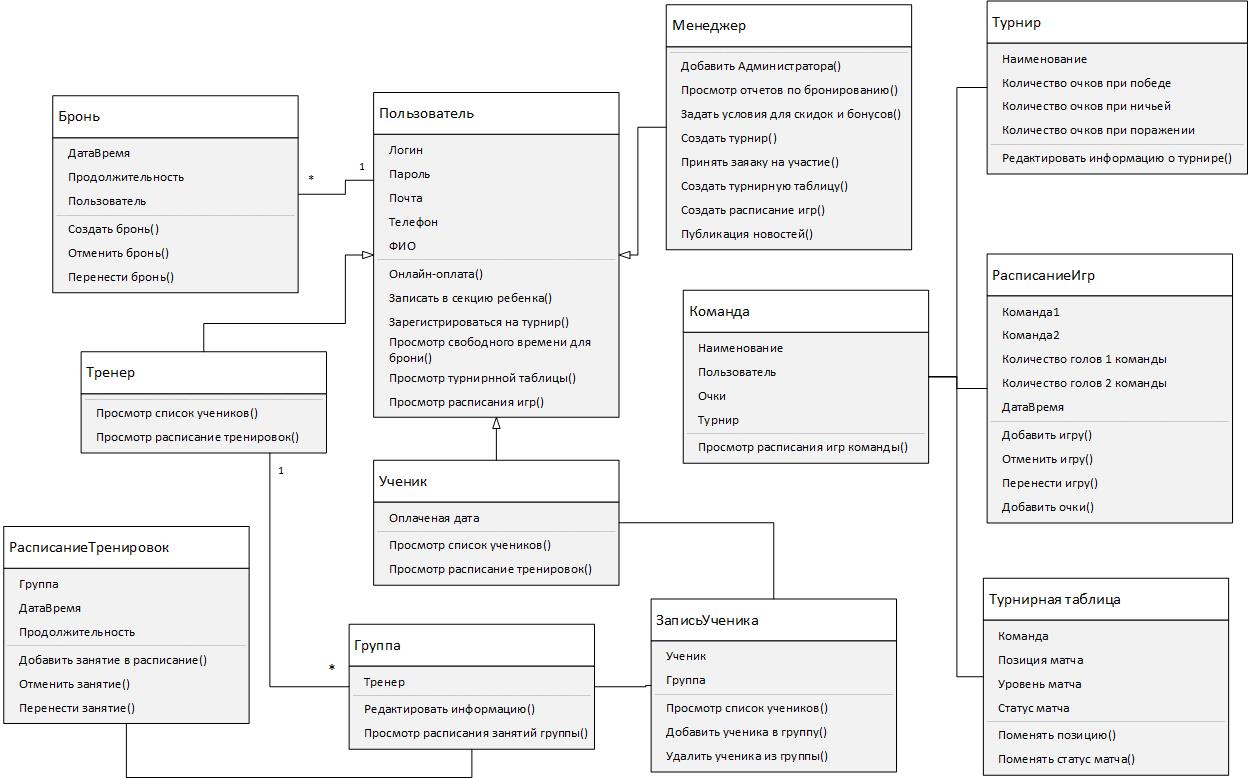
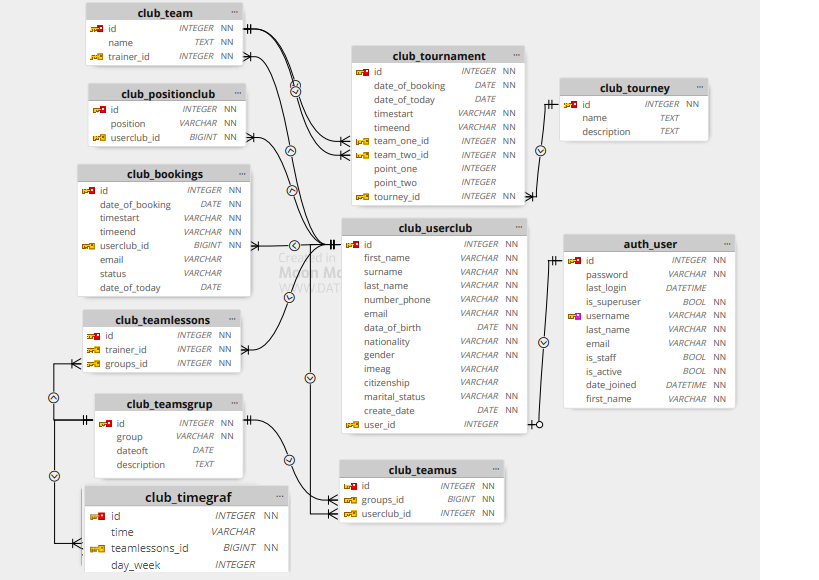


Рис.2.8*.* Диаграмма классов в АС.

* 1. Разработка физической модели базы данных

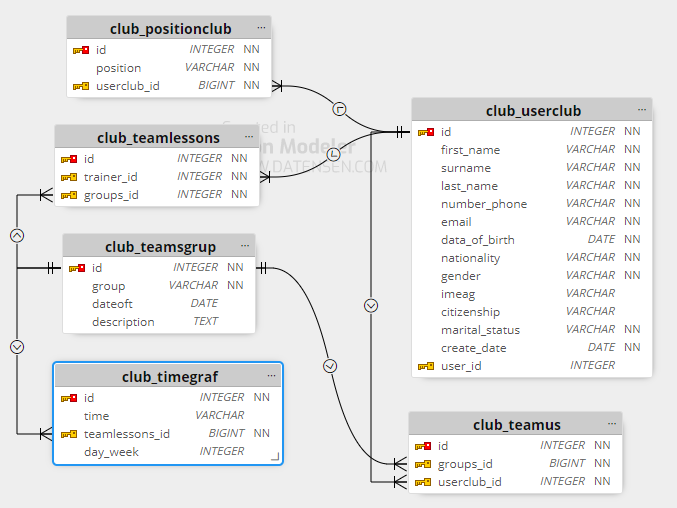
Структура базы данных разработана таким образом, чтобы избежать дублирования данных, обеспечить целостность данных, а также максимально облегчить программирование операций доступа к данным. Всего в базе данных имеется 11 таблицы. На схеме базы данных, представленной на Рис.2.7. отображены только основные таблицы. Из схемы исключены технические таблицы веб-фреймворка Django (таблицы хранения сессий пользователей, миграций базы данных), потому что они являются стандартными и отношения к логике базы данных АС не имеют.

*Рис.2.9.* Физическая модель базы данных.

Таблицы используемые для создания расписания учебных групп

Описание таблиц

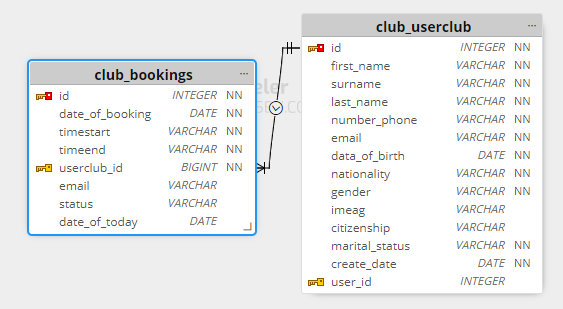
|  |  |
| --- | --- |
| Название | Описание |
| Club\_userclub | Информация о пользователе, таблица справочник |
| Club\_positionclub | Информация о том какой позиции назначен пользователь |
| Club\_teamlessons | информация о том какой тренер назначен учебной группе |
| Club\_teamgruop | информация об учебных группах, таблица справочник |
| Club\_timegraf | информация о том в какое время будет занятие учебной группы |
| Club\_teamus | информация о том какой учебной группе назначен ученик |



*Рис.2.10.* Физическая модель базы данных для учебных групп.

Таблицы, используемые для создания бронирований

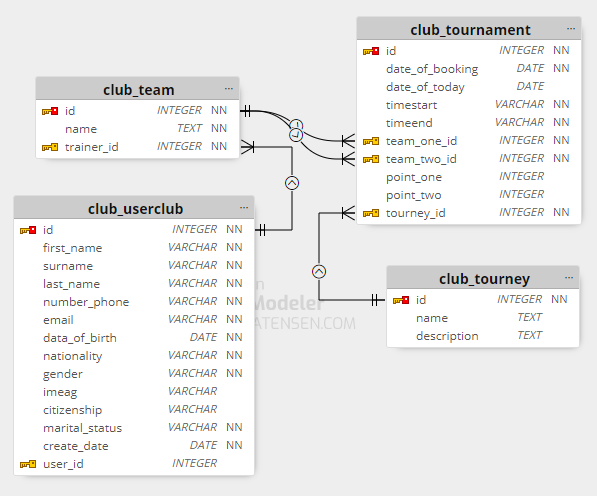
|  |  |
| --- | --- |
| Название | Описание |
| club\_userclub | Информация о пользователе, таблица справочник |
| club\_bookings | Информация о бронированиях |



*Рис.2.11.* Физическая модель базы данных для бронирования.

Таблицы, используемые для создания игр турнира

|  |  |
| --- | --- |
| Название | Описание |
| club\_userclub | Информация о пользователе, таблица справочник |
| club\_team | Информация о командах, таблица справочник |
| club\_tournament | Информация о том какие команды участвуют в игре |
| club\_tourney | Информация о турнирах, таблица справочник |



*Рис.2.12.* Физическая модель базы данных для создания игр турнира.

* 1. Разработка диаграммы компонентов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

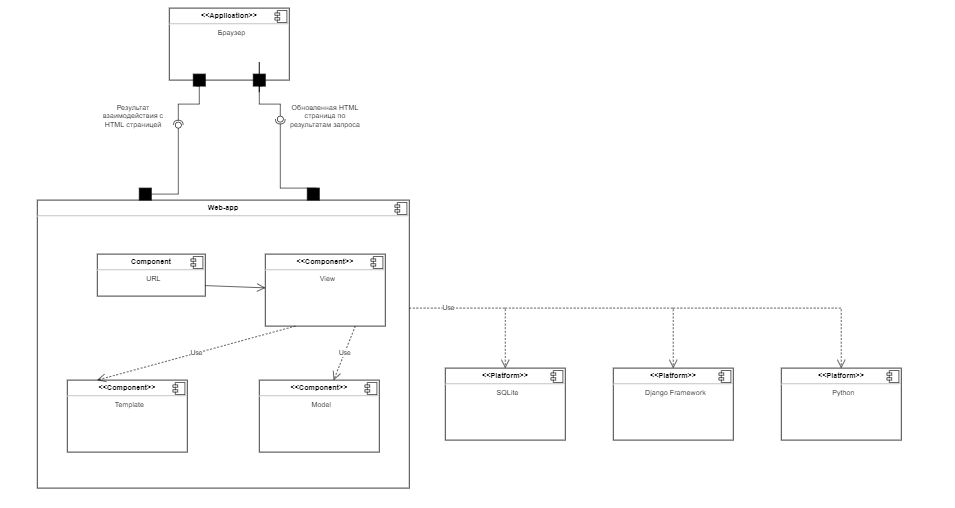


Рис.2.13. Диаграмма компонентов веб-приложения.

* 1. Разработка диаграммы развертывания

Диаграмма развёртывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются “трехмерные” обозначения: узлы системы обозначаются кубиками.

Веб-приложения часто требуют для своей работы некоторой ИТ-инфраструктуры, хранят информацию в базах данных, расположенных где-то на серверах компании, вызывают веб-сервисы, используют общие ресурсы и т. д. В таких случаях полезно иметь графическое представление инфраструктуры, на которую будет развернуто приложение.

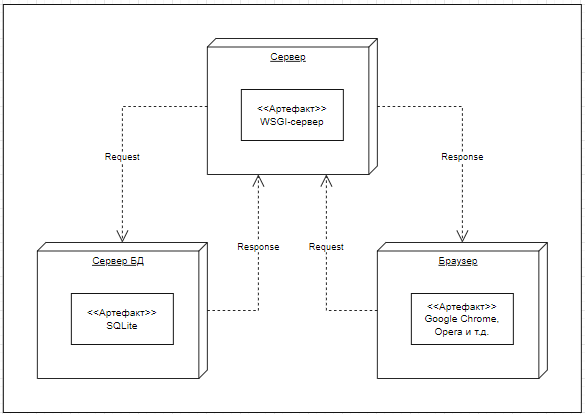


Рис.2.14 Диаграмма развертывания для системы учета для футбольного клуба.

* 1. Выбор средств разработки

В качестве языка разработки выбран язык Python [3] по следующим причинам:

* По сравнению с компилируемым, или строго типизированными языками, такими как С, С++ или Java, язык программирования Python во много раз повышает производительность труда разработчика. Объем программного кода на языке Python обычно составляет треть, или даже пятую часть эквивалентного программного кода на языке С++ или Java, что означает меньший объем ввода с клавиатуры, меньшее количество времени на откладку и меньший объем трудозатрат на сопровождение.
* Кроссплатформенность – язык Python можно применять как под Windows, так и под \*nix системами.
* Наличие большого числа библиотек и инструментов, ускоряющих разработку и повышающих надёжность тестирования программы, что в целом повышает качество поддержки и скорость разработки ПО.

В качестве веб-фреймворка была использована Django [4]— свободный программный каркас для веб-приложений на языке программирования Python, использующий шаблон проектирования MVC (MTV). Ниже представлены возможности, которые повлияли на выбор фреймворка Django для разработки:

* ORM (Object-Relational Mapping), связывание структуры базы данных с описанием таблиц в виде классов для доступа к БД с поддержкой транзакций;
* Расширяемая система шаблонов с тегами и наследованием;
* Подключаемая архитектура приложений, которые можно устанавливать на любые Django-проекты;
* Авторизация и аутентификация, возможность подключения внешних модулей аутентификации;
* Расширяемая встроенная библиотека для работы с формами;
* Система кеширования;
* Система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в фреймворк фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий и др.;

Компоненты фреймворка Django слабо связаны между собой, поэтому их можно достаточно просто заменить на аналогичные. Помимо возможностей, встроенных в ядро Фреймворка, существуют огромное количество пакетов (библиотек), расширяющие его и без того широкие возможности.

В качестве системы управления базами данных применяется СУБД SQLite — компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. Также СУБД SQLite имеет ряд других преимуществ:

Высокая скорость. Благодаря особенностям архитектуры SQLite работает быстро, особенно на чтение. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами.

Надежность. Код на 100% покрыт тестами. Это означает, что протестирован каждый компонент ПО. Поэтому SQLite считается надежной СУБД с минимальным риском непредсказуемого поведения.

Нулевая конфигурация. Перед использованием СУБД не нужна сложная настройка или длительная установка. Для решения большинства задач ей можно пользоваться «из коробки», без установки дополнительных компонентов.

Малый размер. Полностью сконфигурированный SQLite со всеми настройками занимает меньше 400 Кб. Если использовать СУБД без дополнительных компонентов, размер можно уменьшить до 250 Кб. Он зависит только от количества загруженной информации. Несмотря на малый размер, SQLite поддерживает большинство функций стандарта SQL2 и имеет ряд собственных.

Доступность. SQLite находится в публичном доступе. На ее использование нет правовых ограничений, а владельцем считается общество. Можно открывать, просматривать и изменять исходный код установленного ПО.

Кроссплатформенность. СУБД подходит для WINDOWS-подобных систем, MacOS и Windows.

1. РАЗРАБОТКА ДОКУМЕНТАЦИИ
   1. Требования к аппаратным и программным средствам.

Требования сервера ограничиваются требованиями СУБД SQLite 3 и веб-сервера Apache используемой в системе:

* Тип процессора: Процессор Intel CORE i3 и выше.
* Быстродействие процессора: минимум 2 ГГц. Рекомендуется 3 ГГц и выше
* ОЗУ: минимум 8 ГБ. Рекомендуется 16 ГБ или больше. Максимум: максимальный объем, поддерживаемый операционной системой.
* Свободного дискового пространства для базы данных – 100 ГБ, так как со временем размер базы данных будет увеличиваться.

* + 1. Требования к клиентской части.

Клиент должен использовать один из следующих веб браузеров:

* Google Chrome,
* Mozilla Firefox,
* Yandex Browser,
* Opera;
* Internet Explorer 10 версии или выше;
* Safari 5 или выше;
  + 1. Требования к серверной части.

Сервер должен удовлетворять следующим критериям:

* Операционная система семейства Windows (GNU/Linux);
* Веб-сервер Apache с установленной на нем nginx и uWSGI.
* Интерпретатор Python 3.x c установленным пакетным менеджером PIP и менеджером виртуальных окружений Virtualenv.
* Система управления базами данных SQLite 3.0 и выше.
  1. Руководство программиста
     1. Средства реализации программы

Система реализована на языке программирования Python 3.9.2 с использованием веб-фреймворка Django 4.0, реализующий паттерн MVC(MTV). В качестве СУБД система использует SQLite 3.0.

* + 1. Входные и выходные данные

Входные данные (вводимые пользователем) – это данные турниров, данные брони, данные учеников, данные групп, данные занятий, данные о новостях, данные сотрудников.

Выходные данные – это данные турниров, данные брони, данные учеников, данные групп, данные занятий, данные о новостях, информация о введенных данных, все проделанные операции в виде отчетов.

* + 1. Описание установки системы на серверную машину.

Система разрабатывалась и тестировалась на ОС Windows 11 21H2. Дальнейшая инструкция предназначена для ОС Windows 11.

Для быстрой установки всех зависимостей системы, приведённых в таблице 3.1, выполните скрипт pip install <зависимость> из корневой директории проекта с правами суперпользователя.

Также возможна и ручная установка как через терминал (установочные команды можно найти в bash-скриптах в корневой директории проекта), так и через графический интерфейс ОС. Тем не менее, установка через терминал является предпочтительным способом установки.

Таблица 3.1. Список системных зависимостей

|  |  |  |
| --- | --- | --- |
| Название | Описание | Версия |
| Python | Язык программирования, на котором реализована данная система. | 3.x.x (версии 2.х.х не подходят) |
| Pip | Пакетный менеджер, предназначенный для установки сторонних модулей и библиотек языка программирования Python. | Последняя доступная версия |
| Virtualenv | Модуль языка Python для создания виртуального окружения. Позволяет устанавливать пакетные зависимости приложений, отдельно для каждого проекта. | Последняя доступная версия |

**Установка и запуск системы**

Здесь приводится инструкция по ручной установке библиотек и программных зависимостей и запуску системы.

В терминале OC активируйте виртуальное окружение командой терминала source **<путь до директории виртуального окружения> bin/activate.** Из корневой директории системы перейдите к директории проекта по команде **cd < путь до\_ директории\_проекта>/club/** ивнутри этой директории вы найдете файл **requirements.txt** в, котором хранятся названия всех зависимостей проекта. Для установки всех зависимостей выполните команду **pip install -r requirements.txt.** Основные проектные зависимости системы представлены в Таблице 3.2.

Таблица 3.2. Основные проектные зависимости системы

|  |  |
| --- | --- |
| **Зависимость** | **Назначение** |
| Django | Надёжный и многофункциональный веб-фреймворк для разработки веб-приложений на языке программирования Python. |
| Pillow | Библиотека для отображения картинок в проекте |
| django-debug-toolbar | Отладочная панель, которая, при запуске проекта позволяет просматривать настройки проекта, SQL запросы, время выполнения программы, трассировку. |
| django-admin | Пакет, расширяющий функциональные возможности вложенных классов администраторской панели (admin-class) |

После создания базы данных и установки зависимостей можно приступить к миграциям. Миграции описывают структуру БД и изменения в ней на разных этапах разработки. В процессе применения миграций в базе данных создаются и изменяются таблицы и связи. Для запуска этого процесса нужно перейти в директорию contest\_root внутри директории проекта и выполнить следующую команду **/manage.py migrate**

После применения миграций в первый раз Django предлагает создать нового администратора проекта. Если этого не произошло, можно создать пользователя вручную и в дальнейшем использовать его логин и пароль для входа на администраторскую часть приложения. Для создания суперпользователя системы введите в терминал команду.**/manage.py createsuperuser.**

Чтобы веб-сервер Apache смог обнаружить так называемые "статичные файлы" – таблицы стилей CSS, шрифты, скрипты JavaScript проекта, необходимо их перенести одну директорию по команде.**/manage.py collectstatic.**

Кроме запуска проекта через Apache, также существует возможность запуска с помощью отладочного сервера встроенного в django. Он предназначен для разработчиков и может не выдержать реальную нагрузку. На сайт можно будет зайти таким же образом, как и в случае запуска на Apache, но через порт 8000. Например, localhost:8000. Команда запуска.**/manage.py runserver 127.0.0.1:8000**

* + 1. Описание ключевых папок проекта

Django предоставляет каркас проекта, состоящий из набора папок, в которых располагаются файлы проекта. Структура проекта состоит из нескольких отдельных директорий, называемых приложениями реализующих работу системы. Общая структура представлена на Рис.3.1

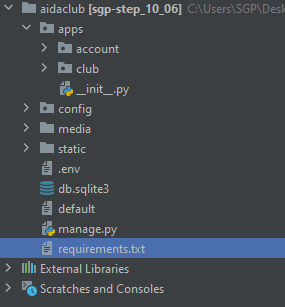


Рис.3.1. Общая структура проекта Django

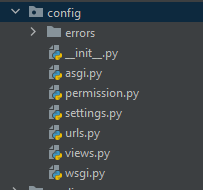


Рис.3.2. Структура приложения Django

Приведем описание файлов уровня приложения:

* Файл **\_\_init\_\_.py** - предназначен для того, чтобы директория проекта Django воспринималась как приложение.
* Файл admin.py - содержит настройки для отображения в администраторской панели.
* Файл **apps.py** - содержит общие настройки приложения.
* Файл **forms.py** - содержит настройки форм приложения
* Файл **tests.py** - содержит наборы тестов для тестирования приложения.
* Файл **urls.py** - содержит список url путей приложения
* Файл **views.py** - содержит представления(контроллеры), которые отвечают бизнес логику приложения и за корректную работу страниц сайта (аналог компонента Controller в модели MVC)
* Файл models.py – содержит классы моделей и методы обработки данных моделей. Модели являются классами, описывающими структуру таблицы базы данных, что позволяет нам работать с таблицы, как с объектами (экземплярами классов).
* Директория migrations – содержит файлы миграций изменений базы данных, в которых описаны изменения структуры базы данных
* Директория templates содержит файлы шаблонов html-страниц уровня приложения.

Приложения **club** которое содержит основные настройки проекта.

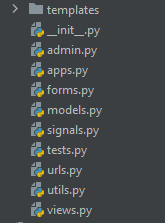


Рис.3.3.Структура приложения Django.

В приложение **club** находятся следующие файлы:

* Файл \_\_init\_\_.py предназначен для того, чтобы директория проекта Django воспринималась как приложение.
* Файл email\_info.py содержит настойки электронной почты.
* Файл settings.py содержит основные настройки и константы проекта.
* url.py содержит url пути уровня проекта.
* wsgi.py предназначен для подключения проекта к веб-серверу Apache.

* 1. Руководство пользователя
     1. Главная страница

При открытии сайт системы всем посетителям отображается главная страница, где находятся меню для направления пользователя на личный кабинет и подобные информации.

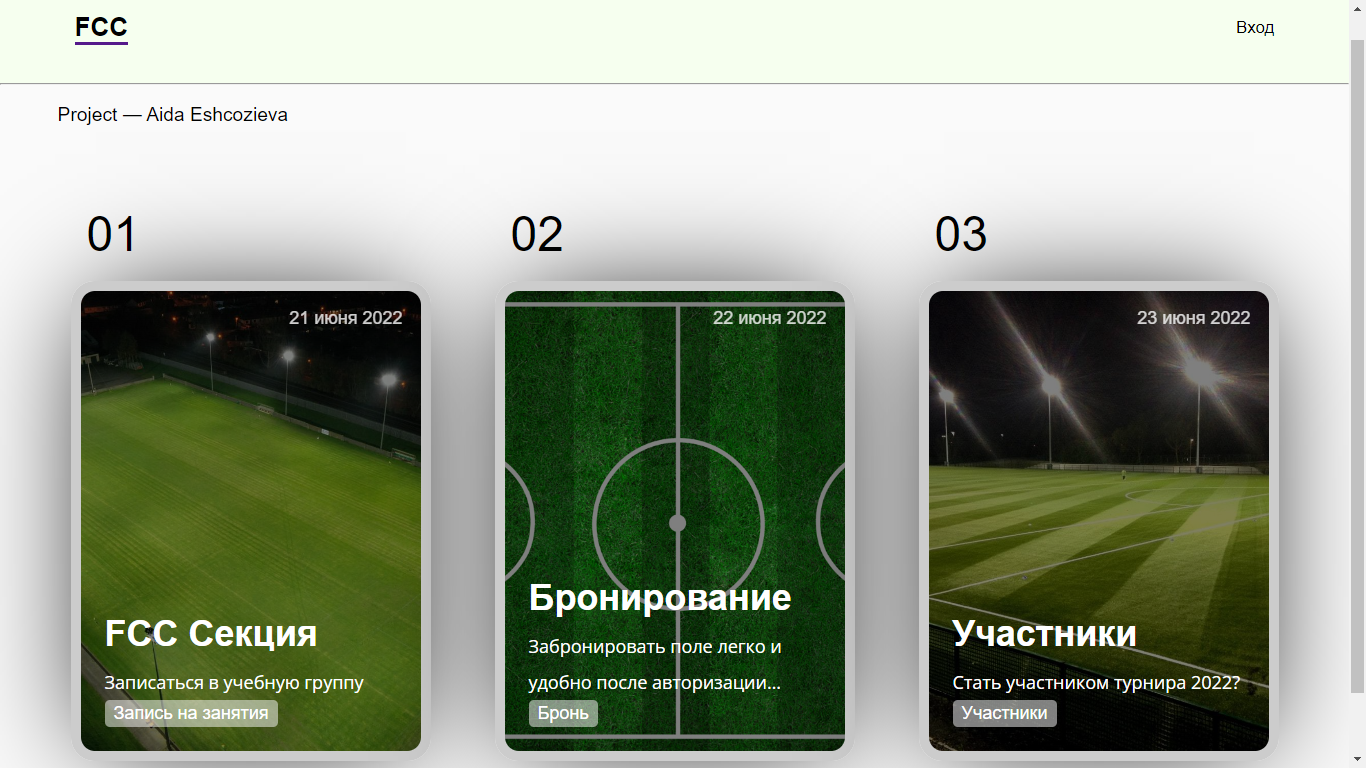


Рис.3. 4. Главная страница.

* + 1. Регистрация

Для регистрации в систему необходимо указать личные данные, после аккаунт будет создан автоматически

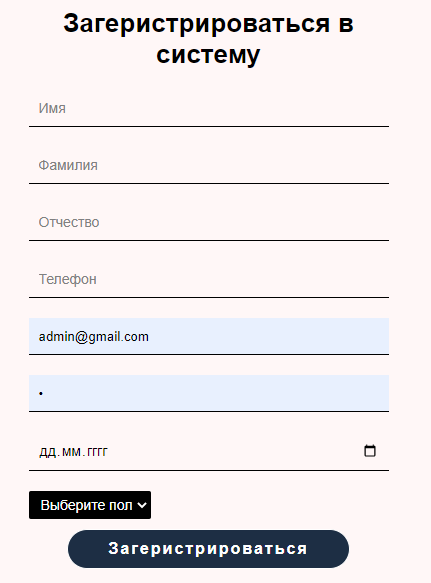
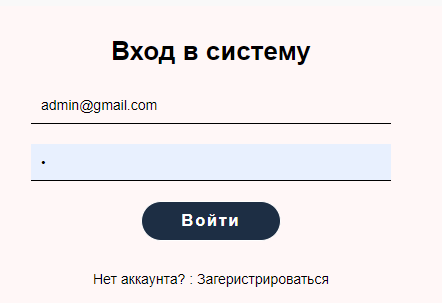


Рис.3.5. Форма регистрации.

* + 1. Авторизация

Для входа в личный кабинет, необходимо ввести личный логин и пароль а при их отсутствии перейти в форму регистрации



*Рис.3.6.* Форма авторизации.

Далее в зависимости от роли пользователя переходит на динамический личный кабинет.

* + 1. Личный кабинет Администратора

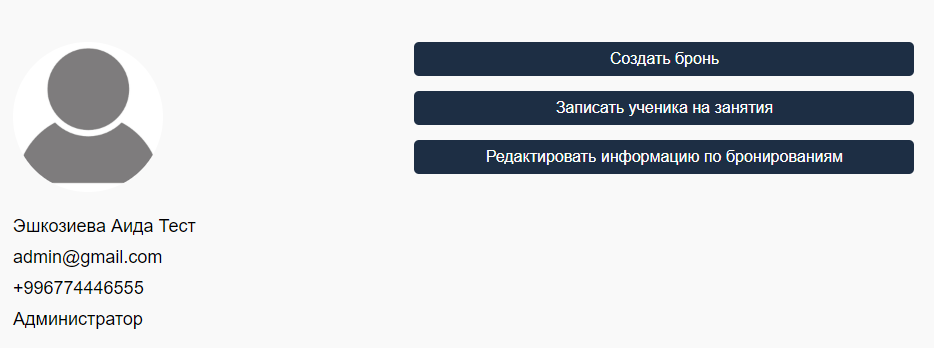


Рис.3.7. Личный кабинет администратора.

В рисунке 3.7. отображен личный кабинет Администратора, у которого есть 3 кнопок.

Для того что бы забронировать поле, необходимо нажать на кнопку “Создать брониь”.

После перейти на форму создания бронирования.



Рис.3.8. Форма брони поля.

В форме брони поля, необходимо указать вышеперечисленные поля, после он автоматически сохраниться в базе данных с id пользователя и его номером, указанным при регистрации.

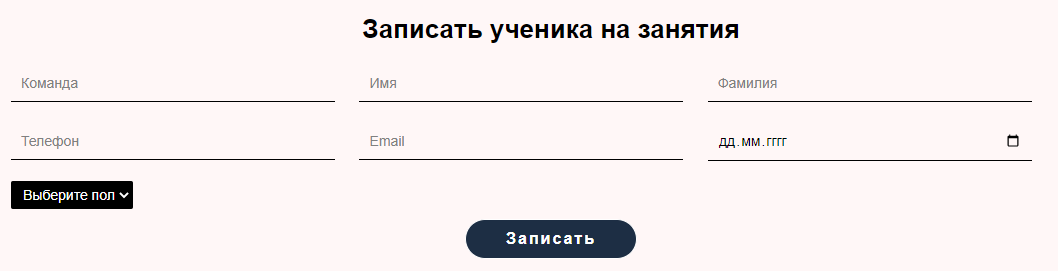


Рис.3.9. Форма записи ученика на занятия.

Форма, где администратор может записать нового ученика в учебную группу.



Рис.3.10. Форма редактировать информацию по бронированиям.

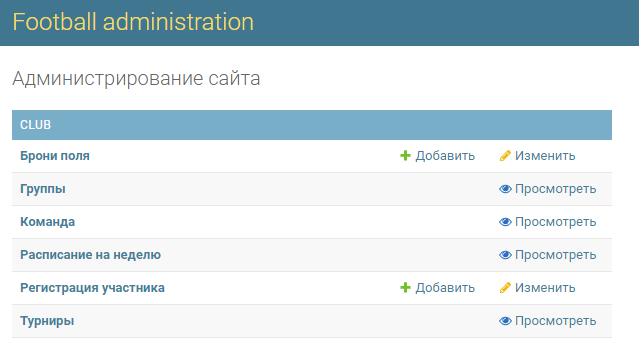


Рис.3.11. Администрирование доступное для Администратора.

* + 1. Личный кабинет Менеджера

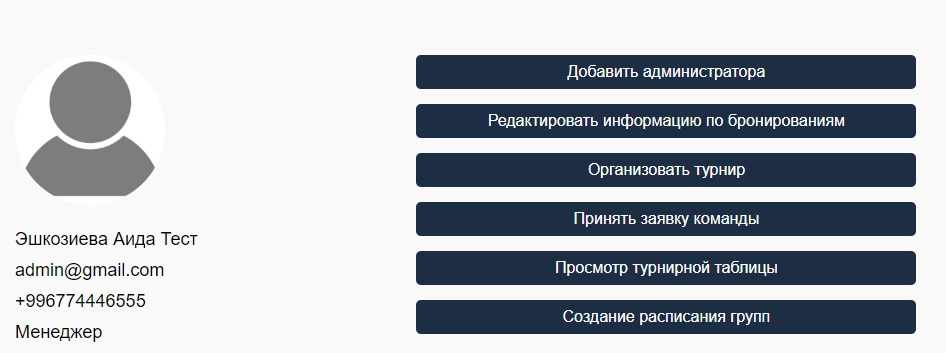


Рис.3.12. Личный кабинет Менеджера.

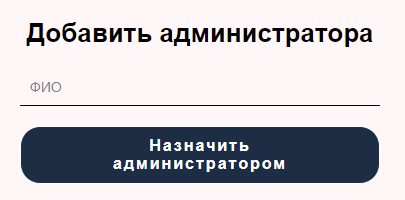


Рис.3.13. Форма назначения администратором пользователя.

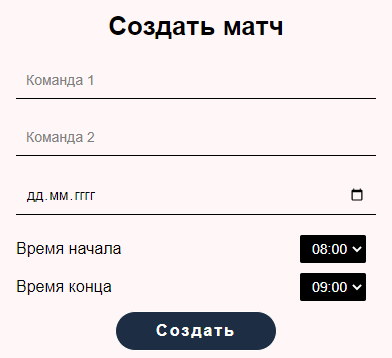


Рис.3.14. Форма создания матч.



Рис.3.15. Форма подтверждения турнира.

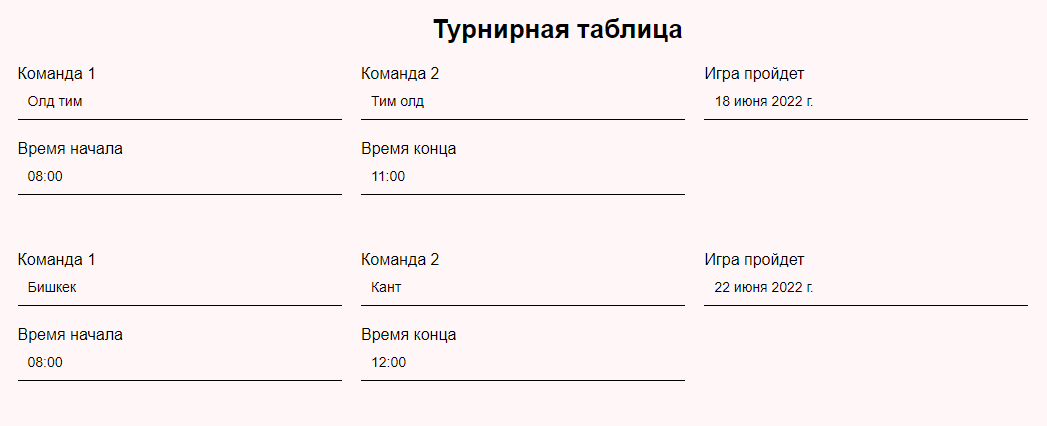


Рис.3.16. Форма просмотра турнирной таблицы.

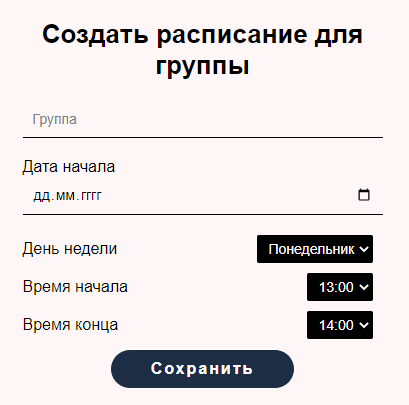


Рис.3.17. Форма создания расписания для групп.

* + 1. Личный кабинет Пользователя

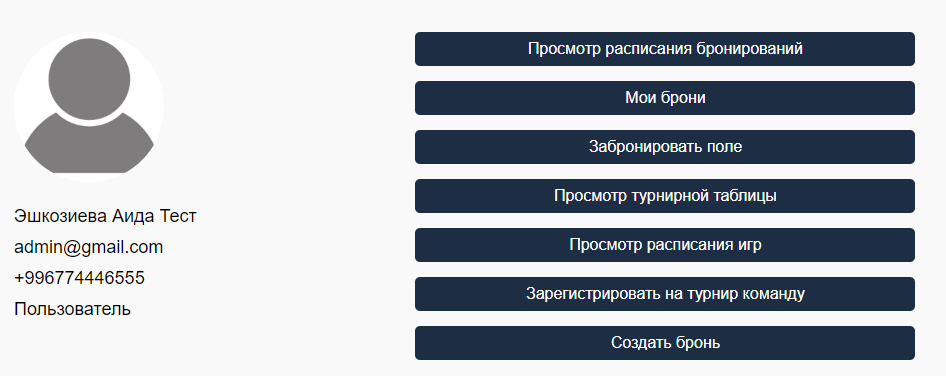


Рис.3.18. Личный кабинет пользователя.

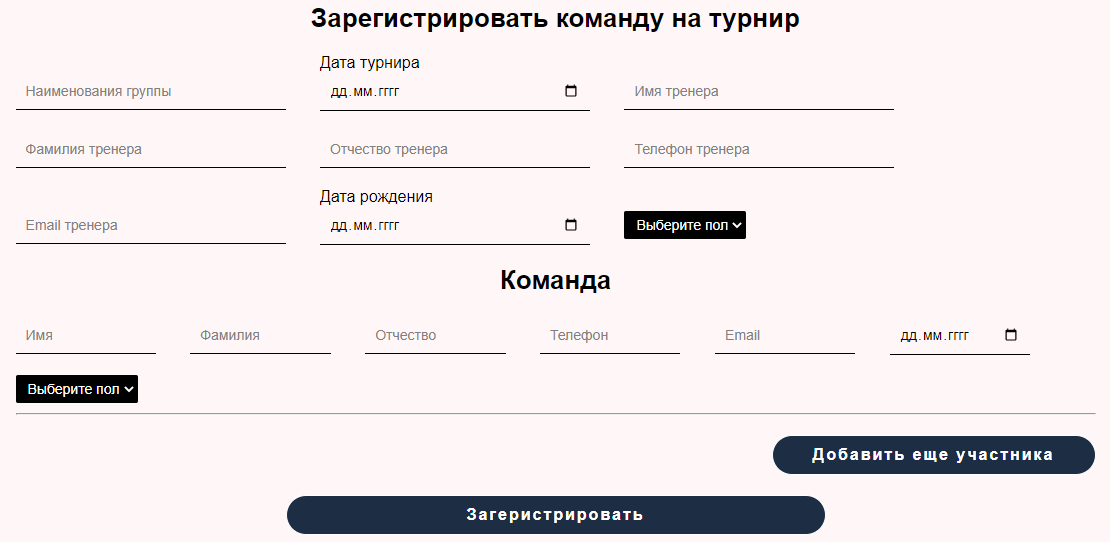


Рис.3.19. Форма регистрации команды на турнир.

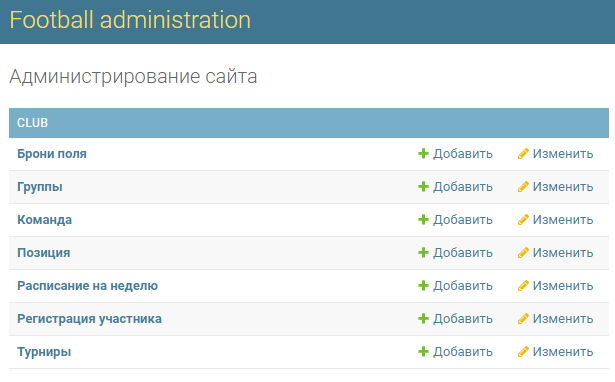


Рис.3.20. Форма администрирования Менеджера

* + 1. Личный кабинет Ученика

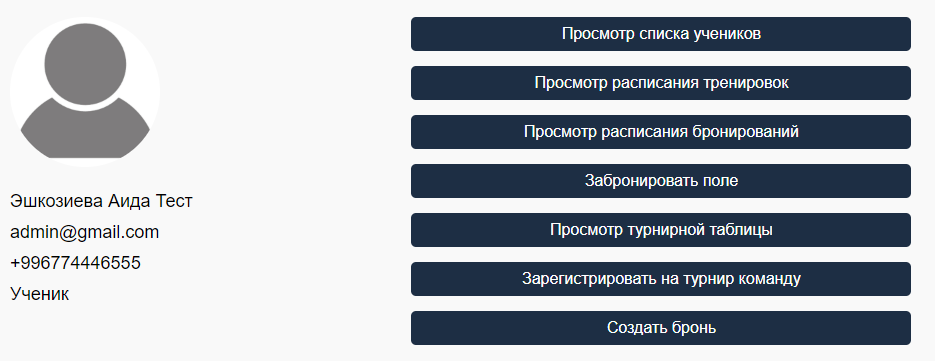


Рис.3.21. Личный кабинет ученика.

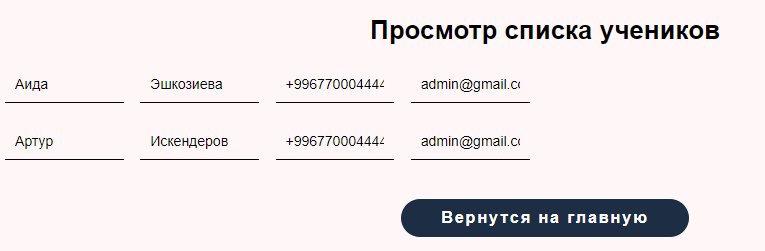


Рис.3.22. Вывод отчета списка учеников.

* + 1. Личный кабинет Тренера

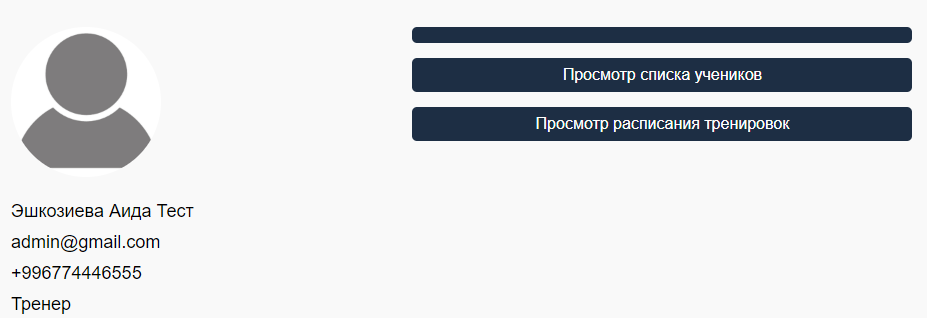


Рис.3.23. Личный кабинет тренера.

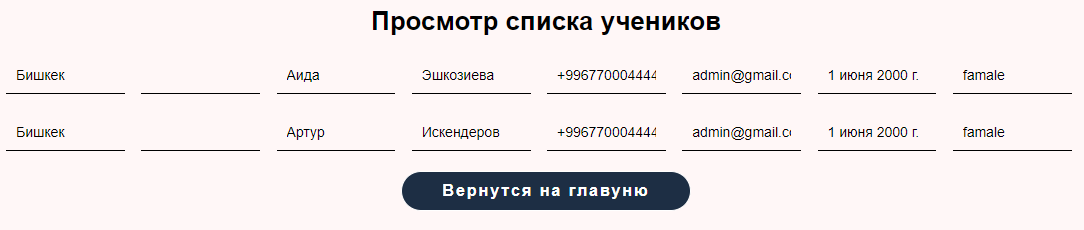


Рис.3.25. Форма просмотра списка учеников

1. ЭКСПЕРИМЕНТАЛЬНЫЙ РАЗДЕЛ
   1. План тестирования
      1. Введение

Этот документ описывает план тестирования для АС “FCC”. Полная стратегия тестирования программного обеспечения состоит из следующих типов испытаний и выполняется в следующем порядке:

1. Тестирование компонентов (модульное тестирование). Тестируются все программные компоненты (при этом проверяется покрытие кода тестами) которые на прямую не взаимодействует с базой данных (в целях сохранности данных). Параллельно проводится анализ кода. Тестирование интерфейсов пользователя;
2. Проверка правильности (валидация). Проводится тестирование программного обеспечения в эмулированной среде производства, чтобы проверить ее функциональные возможности;
3. Приемочные испытания. Проводятся приемочные испытания, чтобы гарантировать, что функциональные возможности ПО соответствуют ожиданиям конечного пользователя.

Приемочные испытания состоят из заключительного набора тестов, которые выполняются перед официальным запуском программной системы.

* + 1. Область тестирования

Цель этого тестирования состоит в том, чтобы определить, насколько хорошо внедряемое программное обеспечение соответствует функциональным требованиям, также идентифицировать обнаруженные проблемы и обеспечить их фиксацию. Тестирование также позволило бы собрать тестовые данные и тестовые результаты для дальнейшего регрессионного тестирования ПО в течение периода его поддержки.

Пожелания пользователей во время работы могут отличаться от их первоначальных; это не должно приниматься во внимание в течение тестирования ПО.

* + 1. Стратегии тестирования

Главное в приемочном испытании — это продемонстрировать, что программное обеспечение и его инфраструктура устойчивы, и работают надежно. Все другие тестирующие действия должны быть закончены перед стартовой фазой приемочного испытания. Приемочное испытание ориентировано на тестирование программного обеспечения с точки зрения пользователя, чтобы проверить, как программное обеспечение используется день ото дня и как оно соответствует определенным требованиям качества. В течение приемочного испытания должен присутствовать представитель от компании клиента.

Дополнительные действия тестирования должны включать: модульное тестирование, тестирование интеграции, безопасности и производительности.

* + 1. Начальные условия (Пререквизиты)

Задачи, которые должны быть решены перед началом тестирующей деятельности:

1. имеется законченная программная спецификация в виде моделей вариантов использования и сценариев использования;
2. работающее программное обеспечение;
3. установленная процедура фиксации обнаруженных проблем в течение испытания;
4. набор определенных вариантов использования для приемочного испытания, чтобы проверить все функциональные возможности программы;
5. выбранная среда тестирования;
6. распределенные тестовые ресурсы,
7. определенные стандарты приемочных испытаний.
   * 1. Методы тестирования

Будут использоваться следующие методы тестирования:

1. Тестовые сценарии – сценарии вариантов использования (с предопределенным вводом и ожидаемыми выходными данными).
2. Тестовые сценарии без данных - тестировщики выберут входные данные во время испытания.
3. Тесты на несанкционированное использование – сценарии действий – попыток получить несанкционированный доступ к данным ПО.
4. Статистика выполнения - набор характеристик работы и сравнение с заданными параметрами.
   * 1. Распределение обязанностей

Определяются следующие роли и обязанности:

1. Лидер проверки качества - человек, ответственный за процесс планирования тестирования и его выполнение.
2. Тестировщик - выполняет действия по тестированию, определенные в плане тестирования.
3. Менеджер по ПО - гарантирует, что тесты выполняются успешно с точки зрения пользователя.
4. Поддержка тестирования - гарантирует, что техническое оборудование на месте и помогает в течение испытаний.
   * 1. Результаты

После тестирования должны быть в наличие следующие документы:

1. План тестирования- этот документ со всеми изменениями, сделанными в ходе процесса тестирования.
2. Запросы на изменение - документ, описывающий изменения ПО, вызванные изменением требований или обнаруженными дефектами в течение испытания.
3. Заключительный отчет, подписанный заказчиком, подтверждающий, что система отвечает всем функциональным требованиям и требованиям качества.
   * 1. Среда тестирования

Следующее программное обеспечение и аппаратная конфигурация должны быть доступны во время тестирования ПО:

1. Сервер: Один компьютер со следующей конфигурацией:
   * Core i5, 3GHz, 4096 MB RAM, 512GB HDD
   * Windows 11
   * PyCharm 2022
   * SQLite 3
   * Подключение к интернету со скоростью не менее 5 Мбит/c
2. Рабочие станции со следующими конфигурациями:
   * Core i3, 2.5GHz, 512 MB RAM
   * Подключение к интернету со скоростью не менее 1 Мбит/c
   * Google Chrome 20 и выше
3. АC FFC конфигурируется следующим образом, добавлением следующих пользователей:
   * Пользователь
   * Тренер
   * Ученик
   * Менеджер
   * Администратор
   1. Сценарии тестирования

Для тестирования были разработаны сценарии. Следующие сценарии тестирования предназначены для проверки программных функций. Часть этих сценариев повторно описывают сценарии вариантов использования.

Каждый сценарий тестов состоит из следующего:

1. Описание тест кейса.
2. Исходные данные для сценария.
3. Процесс тестирования, состоящий из нескольких шагов.
4. Варианты для тестов.
5. Фактическое – ожидаемое.

Ниже приведены некоторые сценарии тестирования:

Тестовый сценарий для авторизации

**Описание:** Пользователь входит в систему.

Исходные данныевводит пользователь.

1.Пользователь переходит по ссылке системы

2.Пользователь вводит логин.

3.Пользователь вводит пароль.

4.Пользователь нажимает кнопку «Войти»

* 1. Функциональное тестирование

После разработки приложения, были проведены модульные тестировании всех частей кода.Пользователь заходит в личный кабинет:

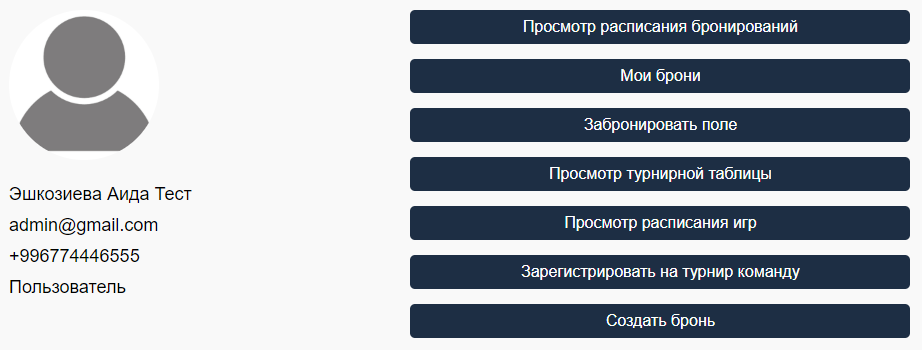


Рис.4.1. Личный кабинет пользователя после авторизации.

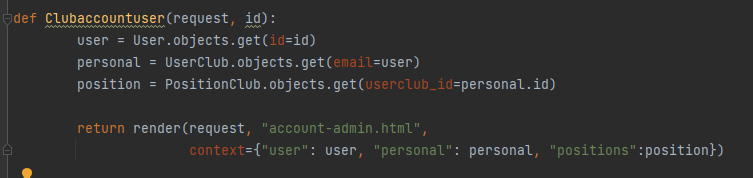


Рис.4.2. Функция отправки данных с БД в личный кабинет.



Рис.4.3. Форма брони.

Как на рис.4.3 можно заметить, после перехода на форму бронирования личного кабинета, отображается поля только даты, времени начала и конца.

Для вывода поступающей информации на консоль, будут выведены такие данные, которые не отображаются в интерфейсе формы бронирования в виде:

* Пользователь
* Номер пользователя
* Дата брони

После отправки данных через форму, функция обрабатывает и сохраняет данные в базу данных и в нашем случае выводит некоторые информации на консоль для проверки:

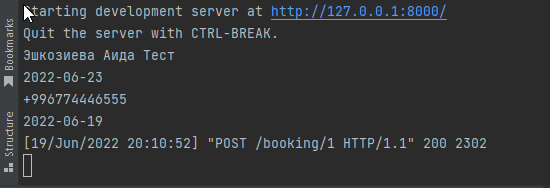


Рис.4.5. Функция приема данных с формы и сохранение в БД.

Как мы можем наблюдать на рис.4.5, данные с формы вывелись в терминал

Далее проверим на наличие данной брони на форме отчетности броней:

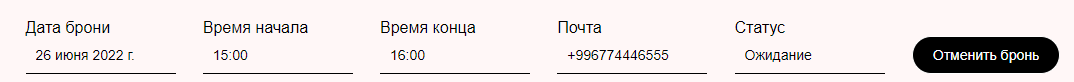


Рис.4.6. Форма отчетности.

Данные успешно сохранились и выводятся на форму со статусом “Ожидание”

* 1. **Модульное (Unit) тестирование**

Сценарий 1. Авторизация в системе под недействительными данными.

Тестовые варианты сценария «Авторизация в системе по ролям».

Всего было проведено 5 тестов.

Пример кода Unit теста:



Рис.4.7. Код Unit теста для сценария 1.

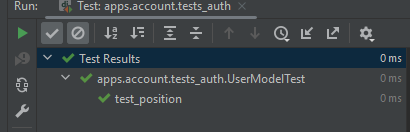


Рис.4.8. Результат тестирования в PyCharm IDEA.

Ожидаемые результаты полностью совпадают с фактическими. Тесты пройдены успешно!

Сценарий 2. Изменение роли пользователя на администратора.



Рис.4.9. Код Unit теста для сценария 2.

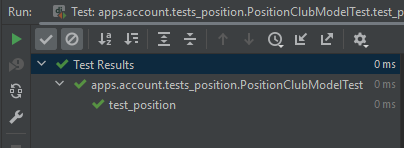


Рис.4.10 Результат тестирования в PyCharm IDEA.

Ожидаемые результаты полностью совпадают с фактическими. Тесты пройдены успешно!

Сценарий 3. Вывод (чтение) данных и проверка валидности выходных данных.



Рис.4.11. Код Unit теста добавления брони теста для сценария 3.

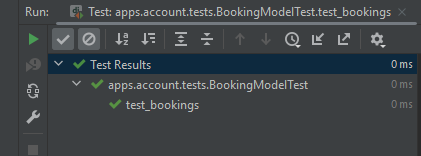


Рис.4.12. Результат тестирования в PyCharm IDEA.

Ожидаемые результаты полностью совпадают с фактическими. Тесты пройдены успешно!

* 1. Расчет метрик кода

Цикломатическая сложность части программного кода — количество линейно независимых маршрутов через программный код. Например, если исходный код не содержит никаких точек ветвления или циклов, то сложность равна единице, поскольку есть только единственный маршрут через код. Если код имеет единственный оператор IF, содержащий простое условие, то существует два пути через код: один если условие оператора IF имеет значение TRUE и один — если FALSE.

На рис.4.13. показан цикломатическая сложность view.py.

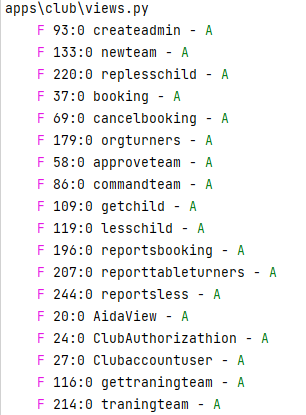


Рис.4.13. Цикломатическая сложность view.py.

Что значат обозначения на рис.4.13. показаны на рис.4.14. - 4.15.

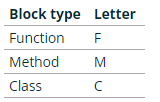
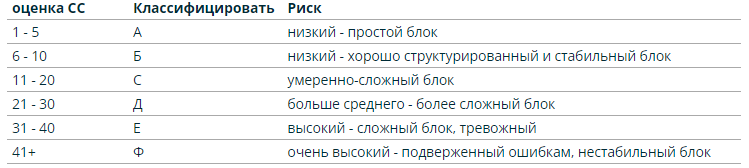


Рис.4.14. Классификация блоков программного кода.



4.15.Оценки по цикломатической сложности.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы была разработана автоматизированной системы учета взаимодействии менеджеров, организации турниров и броней поля футбольного клуба “FCC”. В процессе работы были достигнуты следующие результаты:

* Проведен общий анализ предметной области и существующих аналогов.
* Анализ необходимости разработки данной системы.
* Изучены и проанализированы бизнес-процессы работы в футбольном клубе “FCC”.
* Проанализированы требования заказчика и разработана спецификация требований к системе (Разработана концептуальная модель системы);
* Проведено моделирование будущей системы (построена модель TO-BE в виде диаграммы idef0, диаграмма классов);
* Разработана база данных системы;
* Разработан пользовательский интерфейс системы;
* Разработана веб-ориентированная система, позволяющая выполнять следующие функции:
  + Добавление и редактирование данных о бронированиях и о турнирах.
  + Учет команд и их тренеров.
  + Заполнение данных турнирной таблицы.
  + Создание и редактирование расписания учебных групп.
  + Регистрация сотрудников с разными привилегиями доступа в зависимости от должности персонала.
  + Организация турниров.
* Приведен результат функционального и модульного тестирования, расчет метрик кода.
* Разработано руководство программиста и пользователя.

**Внедрение системы в футбольный клуб «FCC» позволит**:

1. Снять с менеджера непрофильной нагрузки или существенно ее уменьшить, что позволит ему уделять больше времени непосредственно работе по реализации футбольного клуба.
2. Сократить рутинные операции по распределению графика и турнирных таблиц вручную, что позволит существенно уменьшить ошибки, повысить качество и информативность в период проведения занятий и турниров.

Список использованной литературы

1. Салиев А.Б., Стамкулова Г.К., Тен И.Г., Мусина И.Р.,Макиева З.Дж.

Методические указания к выполнению выпускной квалификационной работы бакалавра по направлению 710400 - «Программная инженерия» //Кырг. гос. техн.ун-т, Бишкек, 2016. – 26.

1. Вигерс Карл, Битти Джой

Разработка требований к программному обеспечению. 3е изд., дополненное /Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург,2014. — 736 стр.

1. <https://docs.djangoproject.com/en/4.0/> - официальная документация фреймворка Django на английском.
2. <https://django.fun/docs/django/ru/4.0/> - документация Django 4.0 на русском языке
3. <https://creately.com/blog/ru/диаграмма/учебное-пособие-по-последовательной/>

- Учебное пособие по диаграммам последовательностей

6) <https://metanit.com/python/django/> - руководство по веб-фреймворку Django

ГЛОССАРИЙ

Таблица Г.1. Глоссарий

|  |  |  |
| --- | --- | --- |
| **Термин** | **Сокращение** | **Описание** |
| Автоматизированная система | АС | Единый нормативный документ для учета заболеваемости, причин обращений населения в медицинские учреждения всех ведомств, причин смерти. |
| Object-Relational Mapping (рус. объектно-реляционное отображение) | ORM | Технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных |
| Application programming interface (рус. интерфейс программирования приложений) | API | Набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой) для использования во внешних программных продуктах. |
| Футбольный клуб Чуй | FCC | Система автоматизации оборота документов между пользователями |
| Model-View-Controller (модель-представление-контроллер) | MVC | Схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. |

Продолжение Таблицы Г.1. Глоссарий

|  |  |  |
| --- | --- | --- |
| Система управления базами данных | СУБД | Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных. |
| База данных | БД | Информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств. |
| Фреймворк |  | Программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. |
| Десктопное приложение |  | Это приложения, логика работы которых требует наличия оператора (человека, работающего с программой), содержащие в себе всю полную функциональность и способные работать отдельно на любой машине изолированно от других приложений. |
| Веб-ориентированное приложение |  | [Клиент-серверное](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) приложение, в котором [клиентом](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) выступает [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), а [сервером](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)) — [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). |
| Uniform Resource Locator(рус. Единый указатель ресурса) | URL | Универсальный указатель ресурса, служит стандартизированным способом записи адреса ресурса в сети Интернет. |
| Apache |  | кроссплатформенный HTTP сервер с открытым исходным кодом развиваемый The Apache Software Foundation, |

Продолжение Таблицы Г.1. Глоссарий

|  |  |  |
| --- | --- | --- |
| Каркас веб-приложений |  | Программный каркас, предназначенный для создания динамических веб-приложений, сетевых приложений, сервисов или ресурсов. Он упрощает разработку и избавляет от необходимости написания рутинного кода. Многие каркасы упрощают доступ к базам данных, разработку интерфейса, и также уменьшают дублирование кода |
| Интерпретатор |  | Транслятор, способный параллельно переводить и выполнять программу, написанную на алгоритмическом языке высокого уровня. |
| Bourne again shell | bash | Командный процессор (оболочка), работающий, в интерактивном режиме в текстовом окне. Одна из наиболее популярных современных разновидностей командного процессора в системах WINDOWS. Bash может читать команды из файла, который называется скриптом. |
| Platform as a Service | PaaS | Модель предоставления облачных вычислений, при которой потребитель получает доступ к использованию информационно-технологических платформ: операционных систем, систем управления базами данных, связующему программному обеспечению, средствам разработки и тестирования, размещённым у облачного провайдера. |

ПРИЛОЖЕНИЕ I

Описание таблиц

Общее содержание модели базы данных приведены в таблице ПI.1.

*Таблица* *ПI.1 Описание модели базы данных.*

|  |  |
| --- | --- |
| Название | Описание |
| club\_userclub | Информация о пользователе, таблица справочник |
| club\_positionclub | Информация о том какой позиции назначен пользователь |
| club\_teamlessons | Информация о том какой тренер назначен учебной группе |
| club\_teamgruop | Информация об учебных группах, таблица справочник |
| club\_timegraf | Информация о том в какое время будет занятие учебной группы |
| club\_teamus | Информация о том какой учебной группе назначен ученик |
| club\_bookings | Информация о бронированиях |
| club\_team | Информация о командах, таблица справочник |
| club\_tournament | Информация о том какие команды участвуют в игре |
| club\_tourney | Информация о турнирах, таблица справочник |

*Таблица ПI.2 Описание таблицы «club\_userclub ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| first\_name | varchar(255) | No | Yes |
| surname | varchar(255) | No | Yes |
| last\_name | varchar(255) | No | Yes |
| number\_phone | varchar(128) | No | Yes |

*Таблица ПI.3 Описание таблицы «club\_positionclub ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| position | varchar(255) | No | Yes |
| userclub\_id | bigint | No | Yes |

*Таблица ПI.4 Описание таблицы «club\_teamlessons ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| team | varchar(255) | No | Yes |
| traner | varchar(255) | No | Yes |
| first\_name | varchar(255) | No | Yes |
| surname | varchar(255) | No | Yes |

*Таблица ПI.5 Описание таблицы «club\_teamgruop».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| group | varchar(255) | No | Yes |
| is\_tourney | bool | No | Yes |
| is\_study | bool | No | Yes |
| traner\_id | bigint | No | No |

*Таблица ПI.6 Описание таблицы «* *club\_teamus».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| first\_name | varchar(255) | No | Yes |
| surname | varchar(255) | No | Yes |
| last\_name | varchar(255) | No | Yes |
| number\_phone | varchar(255) | No | Yes |

*Таблица ПI.7 Описание таблицы «* *club\_timegraf ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| weeks | varchar(255) | No | No |
| teamsgrup\_id | bigint | No | Yes |
| time | varchar(255) | No | No |

*Таблица ПI.8 Описание таблицы «* *club\_bookings ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| date\_of\_booking | date | No | Yes |
| timestart | varchar(255) | No | Yes |
| timeend | varchar(255) | No | Yes |
| userclub\_id | bigint | No | Yes |

*Таблица ПI.9 Описание таблицы «* *club\_tournament ».*

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | PrimaryKey | NotNull |
| id | integer | Yes | Yes |
| team\_one\_id | bigint | No | Yes |
| team\_two\_id | bigint | No | Yes |
| timestart | varchar(255) | No | Yes |
| tourney\_id | bigint | No | No |

ПРИЛОЖЕНИЕ II

Листинг программы (серверная часть)

Views.py

from \_\_future\_\_ import division

from django.db.models import Q

from django.views.generic import ListView,DetailView

from django.shortcuts import redirect, render

from .models import \*

from django.contrib.auth.mixins import LoginRequiredMixin

from django.contrib.auth.decorators import login\_required, user\_passes\_test

from django.contrib.auth.models import User, Group

from datetime import date

import datetime

import json

from django.http.response import HttpResponse

class StaffListView(LoginRequiredMixin,ListView):

model = UserClub

template\_name = 'list.html'

context\_object\_name = 'employee'

paginate\_by = 20

def AidaView(request):

return render(request,"index.html")

@login\_required

def ClubAuthorizathion(request):

return render(request,"account-admin.html")

def Clubaccountuser(request, id):

user = User.objects.get(id=id)

personal = UserClub.objects.get(email=user)

position = PositionClub.objects.get(userclub\_id=personal.id)

return render(request, "account-admin.html",

context={"user": user, "personal": personal, "positions":position})

def booking(request, id):

if request.method == "GET":

user = User.objects.get(id=id)

personal = UserClub.objects.get(email=user)

return render(request,"booking.html", context={"user": user, "personal": personal})

if request.method == "POST":

sss = int(request.POST.get('user-id'))

obj = UserClub.objects.get(id=sss)

date\_of\_booking = request.POST.get('dateofbooking')

timestart = request.POST.get('time')

number = obj.number\_phone

date\_of\_today = date.today()

status = "Ожидание"

Bookings.objects.create(userclub=obj, date\_of\_booking=date\_of\_booking,timestart=timestart,email=number,status=status, date\_of\_today=date\_of\_today )

return render(request, "index.html")

return render(request, "booking.html")

def approveteam(request):

if request.method == "GET":

obj = TeamsGrup.objects.all()

objs = TeamUs.objects.all()

return render(request, "approve-team.html", {"obj":obj,"objs":objs })

return render(request,"approve-team.html")

def cancelbooking(request, id):

if request.method == "GET":

user = User.objects.get(id=id)

personal = UserClub.objects.get(user=id)

bookings = personal.bookings.latest("id")

return render(request, "cancel-booking.html", context={"user": user, "personal": personal,"bookings":bookings})

if request.method == "POST":

sss = int(request.POST.get('user-id'))

obj = UserClub.objects.get(id=sss)

bookings = Bookings.objects.get(id=obj.id)

bookings.delete()

return render(request, "index.html")

return render(request,"cancel-booking.html")

def commandteam(request):

if request.method == "GET":

obj = TeamsGrup.objects.all()

objs = TeamUs.objects.all()

return render(request, "command-team.html", {"obj": obj, "objs": objs})

return render(request,"command-team.html")

def createadmin(request):

if request.method == "GET":

obj = UserClub.objects.all()

return render(request,"create-admin.html", {'obj':obj})

if request.method == "POST":

fname = request.POST.get('admins')

obj = UserClub.objects.all()

for i in obj:

if i.full\_name == fname:

objs = UserClub.objects.get(id=i.id)

pcog = PositionClub.objects.get(userclub\_id=objs.id)

pcog.position = "Администратор"

pcog.save()

return render(request, "create-admin.html")

def getchild(request):

if request.method == "GET":

obj = TeamLessons.objects.all()

return render(request, "get-child.html", {"obj":obj})

return render(request,"get-child.html")

def gettraningteam(request):

return render(request,"get-traning-team.html")

def lesschild(request):

if request.method == "POST":

team = request.POST.get('team')

traner =request.POST.get('traner')

first\_name = request.POST.get('first\_name')

surname =request.POST.get('surname')

number\_phone = request.POST.get('number\_phone')

email =request.POST.get('email')

data\_of\_birth = request.POST.get('data\_of\_birth')

gender = request.POST.get('gender')

TeamLessons.objects.create(team=team, traner=traner, first\_name=first\_name, surname=surname, number\_phone=number\_phone,email=email,data\_of\_birth=data\_of\_birth,gender=gender)

return render(request, "index.html")

return render(request,"less-child.html")

def newteam(request):

if request.method == "POST":

group = request.POST.get('group')

dateoft = request.POST.get('dateoft')

first\_name =request.POST.get('first\_name')

surname = request.POST.get('surname')

last\_name = request.POST.get('last\_name')

number\_phone = request.POST.get('number\_phone')

email = request.POST.get('email')

data\_of\_birth = request.POST.get('data\_of\_birth')

gender =request.POST.get('gender')

TeamsGrup.objects.create(group=group,dateoft=dateoft,first\_name=first\_name,surname=surname,last\_name=last\_name,number\_phone=number\_phone,email=email,data\_of\_birth=data\_of\_birth,gender=gender)

obj = TeamsGrup.objects.latest('id')

first\_name = request.POST.get('first\_name0')

surname = request.POST.get('surname0')

last\_name = request.POST.get('last\_name0')

number\_phone = request.POST.get('number\_phone0')

email = request.POST.get('email0')

data\_of\_birth = request.POST.get('data\_of\_birth0')

gender = request.POST.get('gender0')

TeamUs.objects.create(groups=obj, first\_name=first\_name, surname=surname,

last\_name=last\_name, number\_phone=number\_phone, email=email,

data\_of\_birth=data\_of\_birth, gender=gender)

if first\_name != "":

o = 0

for o in range(1, 15):

first\_name = request.POST.get('first\_name' + str(o))

testing = None

if first\_name != testing:

first\_name = request.POST.get('first\_name' + str(o))

surname = request.POST.get('surname' + str(o))

last\_name = request.POST.get('last\_name' + str(o))

number\_phone = request.POST.get('number\_phone' + str(o))

email = request.POST.get('email' + str(o))

data\_of\_birth = request.POST.get('data\_of\_birth' + str(o))

gender = request.POST.get('gender' + str(o))

TeamUs.objects.create(groups=obj, first\_name=first\_name, surname=surname,

last\_name=last\_name, number\_phone=number\_phone, email=email,

data\_of\_birth=data\_of\_birth, gender=gender)

return render(request, "index.html")

return render(request,"new-team.html")

def orgturners(request):

if request.method == "GET":

return render(request, "org-turners.html")

if request.method == "POST":

team\_one = request.POST.get('team-one')

team\_two = request.POST.get('team-two')

date\_of\_booking = request.POST.get('dateofbooking')

timestart = request.POST.get('time')

timeend = request.POST.get('time-end')

date\_of\_today = date.today()

Tournament.objects.create(team\_one=team\_one, date\_of\_booking=date\_of\_booking, timestart=timestart, date\_of\_today=date\_of\_today, timeend=timeend,

team\_two=team\_two)

return render(request, "index.html")

return render(request,"org-turners.html")

def reportsbooking(request):

if request.method == "GET":

status = "Актив"

status1 = "Ожидание"

obj = Bookings.objects.filter(Q(status=status))

obj1 = Bookings.objects.filter(Q(status=status1))

return render(request, "reports-booking.html", {"obj":obj, "obj1":obj1})

return render(request,"reports-booking.html")

def reporttableturners(request):

if request.method == "GET":

obj = Tournament.objects.all()

return render(request, "report-table-turners.html", {"obj":obj})

return render(request,"report-table-turners.html")

def traningteam(request):

return render(request,"traning-team.html")

def replesschild(request):

if request.method == "GET":

obj = TeamsGrup.objects.all()

return render(request,"repless-child.html", {'obj':obj})

if request.method == "POST":

sss = request.POST.get('names')

obj = TeamsGrup.objects.all()

for i in obj:

if i.group == sss:

objs = TeamsGrup.objects.get(id=i.id)

dateofn =request.POST.get('dateofn')

dateofc =request.POST.get('dateofc')

weeks = request.POST.get('weeks')

time =request.POST.get('time')

timeend =request.POST.get('time-end')

TimeGraf.objects.create(teamlessons\_id=objs.id,weeks=weeks ,dateofn=dateofn, dateofc=dateofc, time=time,timeend=timeend )

return render(request, "index.html")

return render(request,"repless-child.html")

def reportsless(request):

if request.method == "GET":

obj = TimeGraf.objects.all()

return render(request, "reports-less.html", {'obj': obj})

return render(request,"reports-less.html")

def editbookings(request):

if request.method == "GET":

status = "Актив"

status1 = "Ожидание"

obj = Bookings.objects.filter(Q(status=status))

obj1 = Bookings.objects.filter(Q(status=status1))

return render(request, "editbookings.html", {"obj":obj, "obj1":obj1})

return render(request,"editbookings.html")

def addrovebooking(request, id=None):

if request.method == "GET":

obj = Bookings.objects.get(id=id)

return render(request, "addrovebooking.html", {"obj":obj})

if request.method == "POST":

objid = request.POST.get('user-id')

obj = Bookings.objects.get(id=objid)

sets = request.POST.get('sets')

if sets == "Подвердить":

saveb = obj

saveb.status = "Актив"

saveb.save()

elif sets == "Отказать":

saveb = obj

saveb.status = "Отказ"

saveb.save()

status = "Актив"

status1 = "Ожидание"

obj = Bookings.objects.filter(Q(status=status))

obj1 = Bookings.objects.filter(Q(status=status1))

return render(request, "editbookings.html", {"obj":obj, "obj1":obj1})

return render(request,"addrovebooking.html")

def mybookings(request, id=None):

if request.method == "GET":

users = User.objects.get(id=id)

objo = ""

objs = UserClub.objects.get(email=users)

objo = Bookings.objects.filter(Q(userclub=objs))

return render(request, "mybookings.html", {"obj": objo})

def cancelbooking(request, id=None):

if request.method == "GET":

obj = Bookings.objects.get(id=id)

return render(request, "cancelbooking.html", {"obj":obj})

if request.method == "POST":

objid = request.POST.get('user-id')

obj = Bookings.objects.get(id=objid)

saveb = obj

saveb.delete()

return render(request, "index.html")

return render(request,"cancelbooking.html")

Models.py

from django.db import models

from phonenumber\_field.modelfields import PhoneNumberField

from django.contrib.auth.models import User

POSITION\_CHOICES = [

('Пользователь', 'Пользователь'),

('Администратор', 'Администратор'),

('Менеджер', 'Менеджер'),

('Ученик', 'Ученик'),

('Тренер', 'Тренер'),

]

time\_slots = (

('13:00 - 14:00', '13:00 - 14:00'),

('14:00 - 15:00', '14:00 - 15:00'),

('15:00 - 16:00', '15:00 - 16:00'),

('16:00 - 17:00', '16:00 - 17:00'),

('17:00 - 18:00', '17:00 - 18:00'),

('18:00 - 19:00', '18:00 - 19:00'),

('19:00 - 20:00', '19:00 - 20:00'),

('20:00 - 21:00', '20:00 - 21:00'),

('21:00 - 22:00', '21:00 - 22:00'),

('22:00 - 23:00', '22:00 - 23:00'),

('23:00 - 24:00', '23:00 - 24:00'),

)

time\_slots1 = (

('13:00', '13:00 '),

('14:00', '14:00 '),

('15:00', '15:00 '),

('16:00', '16:00 '),

('17:00', '17:00 '),

('18:00', '18:00 '),

('19:00', '19:00 '),

('20:00', '20:00 '),

('21:00', '21:00 '),

('22:00', '22:00 '),

('23:00', '23:00 '),

)

time\_slots2 = (

('8:00', '8:00 '),

('9:00', '9:00 '),

('10:00', '10:00 '),

('11:00', '11:00 '),

('12:00', '12:00 '),

)

WEEK\_CHOICES = [

('Понедельник', 'Понедельник'),

('Вторник', 'Вторник'),

('Среда', 'Среда'),

('Четверг', 'Четверг'),

('Пятница', 'Пятница'),

('Суббота', 'Суббота'),

('Воскресенье', 'Воскресенье'),

]

STATUS\_CHOICES = [

('Ожидание', 'Ожидание'),

('Актив', 'Актив'),

]

STATUS\_CHOICES1 = [

('Неактивен', 'Неактивен'),

('Активен', 'Активен'),

]

OTHER = "OTHER"

GENDER = (

("M", 'мужчина'),

("F", 'женщина'),

(OTHER, "другое")

)

class UserClub(models.Model):

"""

Модель Администратор которая подтягивает данные из внешней API

"""

MARRIED = 'Married'

SINGLE = 'Single'

WIDOWED = 'Widowed'

DIVORCED = 'Divorced'

MARITAL\_STATUS = (

(SINGLE, "не замужем / не женат"),

(MARRIED, "замужем / женат"),

(DIVORCED, "разведен / разведена"),

(WIDOWED, "вдовец / вдова")

)

user = models.OneToOneField(User, on\_delete=models.CASCADE, blank=True, null=True)

first\_name = models.CharField(max\_length=255, verbose\_name="Имя")

surname = models.CharField(max\_length=255, verbose\_name="Фамилия")

last\_name = models.CharField(max\_length=255, verbose\_name="Отчество")

number\_phone = PhoneNumberField(region="KG")

email = models.EmailField()

data\_of\_birth = models.DateField()

nationality = models.CharField(max\_length=255, verbose\_name="Национальность")

gender = models.CharField(choices=GENDER, default="M", max\_length=8, verbose\_name="Пол")

imeag = models.ImageField("Фотография", upload\_to='', blank=True, null=True)

citizenship = models.CharField(verbose\_name="Гражданство",max\_length=255, null=True)

marital\_status = models.CharField(choices=MARITAL\_STATUS, default=SINGLE,

max\_length=8, verbose\_name="Семейный статус", blank=True)

@property

def full\_name(self):

return self.surname + " " + self.first\_name + " " + self.last\_name

def \_\_str\_\_(self):

return self.full\_name

class Meta:

verbose\_name = "Пользователь"

verbose\_name\_plural = "Пользователи"

class PositionClub(models.Model):

userclub = models.ForeignKey(UserClub, on\_delete=models.CASCADE, related\_name="managerclub")

position = models.CharField(max\_length=255, verbose\_name="Позиция", choices=POSITION\_CHOICES)

def \_\_str\_\_(self):

return self.userclub.\_\_str\_\_()

class Meta:

verbose\_name = "Позиция"

verbose\_name\_plural = "Позиции"

class Bookings(models.Model):

userclub = models.ForeignKey(UserClub, on\_delete=models.CASCADE, related\_name="bookings")

date\_of\_booking = models.DateField(verbose\_name="Дата брони")

date\_of\_today = models.DateField(verbose\_name="Забронированная брони", blank=True, null=True)

timestart = models.CharField(max\_length=255, verbose\_name="Время начала", choices=time\_slots1)

email = models.CharField(max\_length=255, verbose\_name="Номер", blank=True, null=True)

status =models.CharField(max\_length=255, verbose\_name="Статус", blank=True, null=True, choices=STATUS\_CHOICES)

def \_\_str\_\_(self):

return self.userclub.\_\_str\_\_()

class Meta:

verbose\_name = "Бронь поля"

verbose\_name\_plural = "Бронирования поля"

class TeamsGrup(models.Model):

group = models.CharField(max\_length=255, verbose\_name="Наименование группы")

traner = models.ForeignKey(UserClub, on\_delete=models.CASCADE, related\_name="teamsgrup",null=True, verbose\_name="Тренер")

is\_tourney = models.BooleanField(verbose\_name="Участие в турнире", default=True)

is\_study = models.BooleanField(verbose\_name="Учебная группа", default=True)

def \_\_str\_\_(self):

return self.group

class Meta:

verbose\_name = "Группа"

verbose\_name\_plural = "Группы"

class Tourney(models.Model):

tourney = models.CharField(max\_length=255, verbose\_name="Турнир")

date\_begin = models.DateField(verbose\_name="Дата начала турнира", blank=True, null=True)

date\_end = models.DateField(verbose\_name="Дата конца турнира", blank=True, null=True)

description = models.TextField(verbose\_name="Описание турнира", blank=True, null=True)

def \_\_str\_\_(self):

return self.tourney + " --- от " + str(self.date\_begin) + " до " + str(self.date\_end)

class Meta:

verbose\_name = "Турнир"

verbose\_name\_plural = "Турниры"

class Tournament(models.Model):

team\_one = models.ForeignKey(TeamsGrup, on\_delete=models.CASCADE, related\_name="tournament",)

team\_two = models.ForeignKey(TeamsGrup, on\_delete=models.CASCADE, related\_name="tournament1",)

tourney = models.ForeignKey(Tourney, on\_delete=models.CASCADE, related\_name="tournament",blank=True)

date = models.DateField(verbose\_name="Дата Турнира", blank=True, null=True)

timestart = models.CharField(max\_length=255, verbose\_name="Время начала")

goal\_one = models.IntegerField(default=0, verbose\_name="Голы команды 1")

goal\_two = models.IntegerField(default=0, verbose\_name="Голы команды 2")

def \_\_str\_\_(self):

return self.team\_one.\_\_str\_\_() + " x " + self.team\_two.\_\_str\_\_()

class Meta:

verbose\_name = "Матч"

verbose\_name\_plural = "Матчи"

class TeamUs(models.Model):

groups = models.ForeignKey(TeamsGrup, on\_delete=models.CASCADE, related\_name="teamus")

userclub = models.ForeignKey(UserClub, on\_delete=models.CASCADE, related\_name="teamus")

first\_name = models.CharField(max\_length=255, verbose\_name="Имя")

surname = models.CharField(max\_length=255, verbose\_name="Фамилия")

last\_name = models.CharField(max\_length=255, verbose\_name="Отчество")

number\_phone = models.CharField(max\_length=255, verbose\_name="Номер телефона")

email = models.CharField(max\_length=255, verbose\_name="Почта")

data\_of\_birth = models.DateField(verbose\_name="Дата рождения", blank=True, null=True)

gender = models.CharField(max\_length=255, verbose\_name="Пол", choices=GENDER )

status = models.CharField(max\_length=255, verbose\_name="Статус", blank=True, null=True, choices=STATUS\_CHOICES1)

def \_\_str\_\_(self):

return self.groups.group + " --- " + self.first\_name + ' ' + self.surname

class Meta:

verbose\_name = "Участник группы"

verbose\_name\_plural = "Участники группы"

class TimeGraf(models.Model):

teamsgrup = models.ForeignKey(TeamsGrup, on\_delete=models.CASCADE, related\_name="timegraf")

time = models.CharField(max\_length=255,verbose\_name="Время начала", blank=True, null=True, choices=time\_slots2)

weeks = models.CharField(max\_length=255,verbose\_name="День недели",default='Понедельник', null=True, choices=WEEK\_CHOICES)

def \_\_str\_\_(self):

return self.teamsgrup.group + ' --- ' + self.weeks + ' --- ' + self.time

class Meta:

verbose\_name = "Расписание на неделю"

verbose\_name\_plural = "Расписание на неделю"

urls.py

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

from .views import \*

urlpatterns = [

path("",AidaView,name='home'),

path("home/",AidaView,name='homes'),

path("authorizathion/",ClubAuthorizathion,name='authorizathion'),

path('accountuser/<int:id>',Clubaccountuser, name='accountuser'),

path('approve-team/', approveteam, name='approve-team'),

path('booking/<int:id>', booking, name='booking'),

path('cancel-booking/<int:id>', cancelbooking, name='cancel-booking'),

path('command-team/', commandteam, name='command-team'),

path('create-admin/', createadmin, name='create-admin'),

path('get-child/', getchild, name='get-child'),

path('get-traning-team/', gettraningteam, name='get-traning-team'),

path('less-child/', lesschild, name='less-child'),

path('new-team/', newteam, name='new-team'),

path('org-turners/', orgturners, name='org-turners'),

path('report-booking/', reportsbooking, name='report-booking'),

path('report-table-turners/', reporttableturners, name='report-table-turners'),

path('traning-team/', traningteam, name='traning-team'),

path('replesschild/', replesschild, name='replesschild'),

path('reportsless/', reportsless, name='reportsless'),

path('editbookings/', editbookings, name='editbookings'),

path('cancelbooking/<int:id>', cancelbooking, name='cancelbooking'),

path('addrovebooking/<int:id>', addrovebooking, name='addrovebooking'),

path('mybookings/<int:id>', mybookings, name='mybookings'),

]

if settings.DEBUG:

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)