**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”  
(УНИВЕРСИТЕТ ИТМО)**

**ЦЕНТР АВТОРИЗОВАННОГО ОБУЧЕНИЯ ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ**

**ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА**

**Разработка Web-приложения для автоматизации планирования посещений для клиентов фитнес-клуба**



Автор Симонова Маргарита Игоревна \_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество) (Подпись)

**Центр авторизованного обучения информационным технологиям**

Наименование программы **«Python-разработчик»**

Руководитель Кузьмин Константин Михайлович \_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

**К защите допустить**

|  |  |  |
| --- | --- | --- |
| *Заместитель директора ЦАО ИТ,*  *к.т.н.* |  | / *Т.В. Зудилова*/ |

Санкт-Петербург, 2024г.

Обучающийся Симонова Маргарита Игоревна Группа 124/12

(Фамилия, И. О.)

Работа принята «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Работа выполнена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Секретарь ИАК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, И. О.) (подпись)

Листов хранения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Демонстрационных материалов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc162616438)

[ВЫПОЛНЕНИЕ РАБОТЫ 5](#_Toc162616439)

[ЗАКЛЮЧЕНИЕ 28](#_Toc162616440)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc162616441)

**ВВЕДЕНИЕ**

Основываясь на своем опыте посещения фитнес-клубов, в качестве клиента, хотелось бы отметить, что периодически я сталкивалась с проблемой слишком большого количества одновременно занимающихся. Перед тем как идти на занятие мне бы хотелось видеть примерное количество желающих прийти в это же время. Решением данной проблемы, на мой взгляд, могло бы стать создание автоматизированной системы планирования посещений зон клуба. Также, вероятно фитнес-клубу тоже бы хотелось иметь такие данные, ведь это бы позволило им распределить нагрузку равномернее, более гибко управлять тарифами. В связи с представленными выше доводами, считаю данную тему актуальной.

Целью дипломной работы является разработка системы для контроля загруженности фитнес-клуба и планирования клиентами времени посещения.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Проанализировать предметную область.
2. Сформулировать функциональные требования к системе и задокументировать их с помощью модели прецедентов (диаграммы вариантов использования).
3. Разработать модель предметной области.
4. Спроектировать алгоритм решения задачи согласно процедурной парадигме.
5. Спроектировать диаграмму последовательностей.
6. Выбрать подход к разработке системы.
7. Спроектировать диаграмму классов.
8. Выбрать средства реализации.
9. Разработать и создать БД.
10. Разработать серверную часть приложения.
11. Протестировать.
12. Разработать способ развертывания системы.
13. Разработать инструкции.

**ВЫПОЛНЕНИЕ РАБОТЫ**

**Анализ предметной области**

*Концепция проекта*

Для администрации фитнес-клуба, а также для их клиентов, система «Training plan» станет информационной системой, которая позволит отслеживать количество посетителей, планирующих визит в определенное время. Для клиентов фитнес-клуба она поможет спланировать посещение в менее нагруженное время. Для фитнес-клуба введение данной системы в эксплуатацию позволит распределить нагрузку, уменьшить пиковое количество посетителей, что позволит увеличить удовлетворенность предоставляемыми услугами у клиентов, а также позволит разработать новые тарифные планы. В условиях пандемии, когда по санитарно-эпидемиологическим нормам посещение бассейна, а также групповых тренировок было ограничено в количестве занимающихся, данная система бы очень пригодилась. Также на некоторые классы занятий количество мест ограничено и запись обычно ведется вручную, что ведет к возможным ошибкам и нарушениям норм, автоматизация процесса позволит повысить качество предоставляемых услуг.

*Типы пользователей и роли*

Администратор – выполняет функции менеджера пользователей, имеет доступ к данным фитнес-клуба.

Менеджер – составляет расписание занятий, просматривает статистику планируемых посещений, имеет возможность посмотреть кто из клиентов планирует посещение.

Клиент клуба – подает заявку на планируемое посещение, просматривает статистику планируемых посещений.

*Ограничения проекта*

Фитнес-клуб может быть сетью - иметь несколько площадок, а может иметь только одну площадку. Зоны групповых занятий постоянны, то есть в фитнес-клубе выделены залы только под групповые занятия, планировать визит в такие зоны без группового занятия нельзя.

**Функциональные требования к системе**

Представлены в таблице 1.

Таблица 1 – Функциональные требования

|  |  |
| --- | --- |
| **Функциональные требования** | **В рамках проекта** |
| 1. Система обеспечивает возможность пользователю заходить в систему со своими учетными данными | 1.1 Создание в базе данных таблицы, отражающей данные пользователя,  где указана роль (менеджер,  администратор, клиент) |
|  | 1.2 Реализация системы авторизации  и аутентификации пользователя |
| 2. Система обеспечивает возможность пользователю сменить пароль или восстановить его в случае потери | 2.1 Реализация функционала сброса старого пароля и ввода нового |
| 3. Система обеспечивает возможность пользователю выбрать из главного меню необходимый функционал | 3.1 Создание на всех формах панели с главным меню |
| 4. Система обеспечивает возможность пользователям просматривать количество клиентов планирующих посетить каждую из зон клуба в течении выбранного дня | 4.1 Создание в базе данных таблицы, отражающей планируемые посещения |
|  | 4.2 Создание формы, отражающей количество планируемых посещений за выбранный день в выбранной локации |
| 5. Система позволяет администратору  управлять учетными данными пользователей | 5.1 Создание формы работы с учетными данными (создание, изменение, удаление) |
|  | 5.2 Реализация для администратора  возможности создания, изменения, удаления учетных записей |
| 6. Система позволяет администратору менять данные о фитнес-клубе | 6.1 Создание в базе данных таблицы  с информацией о фитнес-клубе и таблицы о доступных зонах и групповых занятиях, графике работы, клубных картах |
|  | 6.2 Создание формы работы с данными фитнес-клуба |
| 7. Система позволяет менеджеру  составлять расписание групповых занятий, указывая оптимальное количество присутствующих, удалять (в случае отсутствия записавшихся) и отменять занятия | 7.1 Создание в базе данных таблицы с информацией о расписании занятий с оптимальным количеством присутствующих |
|  | 7.2 Создание формы работы с расписанием |
| 8. Система позволяет клиенту заявлять о своем планируемом посещении | 8.1 Создание в базе данных таблицы, отражающей планируемые посещения клиентами |
|  | 8.2 Обеспечение возможности создания, удаления планируемых посещений |
|  | 8.3 Создание форм для создания, удаления планируемых посещений |
| 9. Система позволяет менеджеру контролировать загруженность фитнес-клуба | 9.1 Обеспечение возможности просмотра списка планирующих посещение в выбранный день и в выбранной локации |

**Диаграмма вариантов использования**

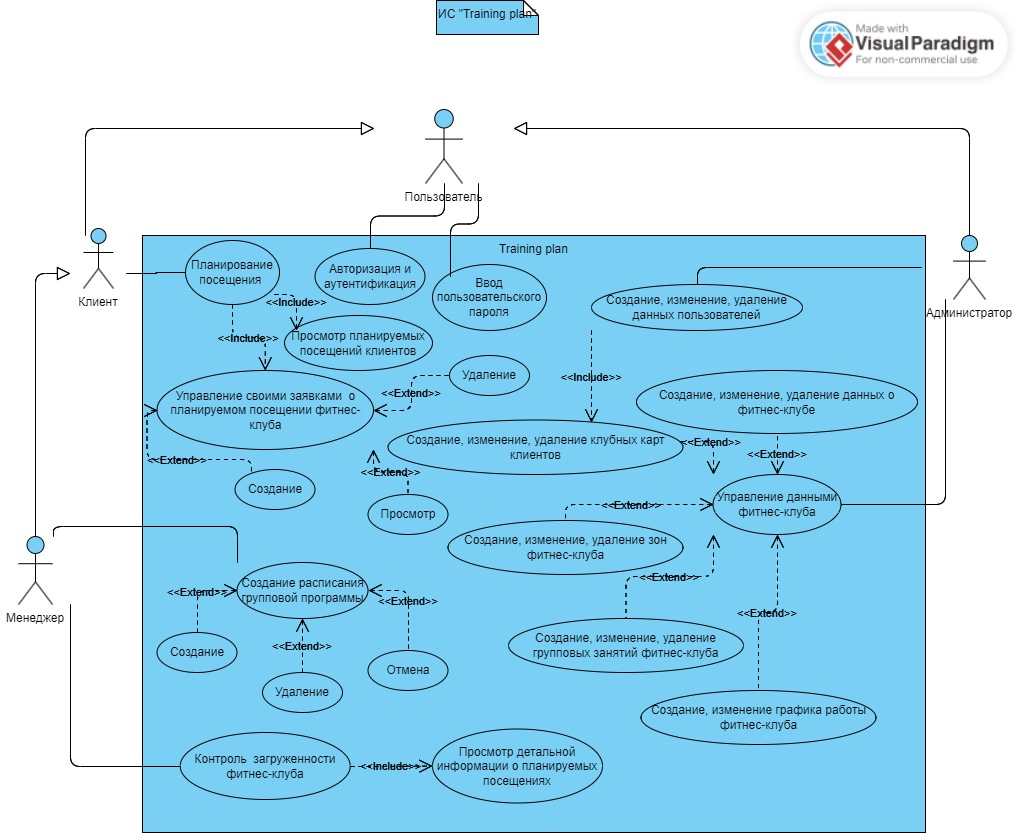
Представлена на рисунке 1. 

Рисунок 1 – Диаграмма вариантов использования

**Описание вариантов использования**

***Сценарий 1 – Авторизация и аутентификация***

Вариант использования: авторизация и аутентификация пользователя в системе

Участники: пользователь

Цель: авторизация и аутентификация

Краткое описание: пользователь вводит логин, пароль, открывается форма, соответствующая роли пользователя

Типичный ход событий:

1. Пользователь вводит логин, пароль

*Исключение 1 Введенный логин не существует*

*Исключение 2 Введенный пароль неверный*

Открывается форма, соответствующая роли пользователя

Раздел исключений:

*Исключение 1 Введенный логин не существует*

1. Пользователю выводится сообщение, что такого логина не существует, предлагается ввести данные снова
2. Пользователь вводит логин, пароль до тех пор, пока логин и пароль не будут легитимными или закрывает форму

*Исключение 2 Введенный пароль неверный*

1. Пользователю выводится сообщение, что пароль неверный, предлагается ввести пароль снова

2. Пользователь вводит пароль до тех пор, пока пароль не будет легитимным или закрывает форму

***Сценарий 2 – Ввод пользовательского пароля***

Вариант использования: ввод нового пароля

Участники: пользователь

Цель: создание своего собственного пароля, смена пароля в случае его утери

Краткое описание: пользователь сбрасывает старый пароль, создает его заново

Типичный ход событий:

1. В случае если необходимо сменить пароль, пользователь вводит свой логин, выбирает сброс пароля

*Исключение 1 Введенный логин не существует*

1. Пользователю предлагается ввести свою почту, если почта в профиле и введенный адрес совпадают, пользователю приходит письмо со ссылкой на сброс пароля

*Исключение 2 Введенный адрес почты не совпадает с указанным в профиле*

1. Пользователь вводит новый пароль, подтверждает его

*Исключение 3 Введенные пароли не совпадают*

1. Пользователю предлагается войти в систему с новыми учетными данными

Раздел исключений:

*Исключение 1 Введенный логин не существует*

1. Пользователю выводится сообщение, что такого логина не существует, предлагается ввести данные снова
2. Пользователь вводит логин, пароль до тех пор, пока логин и пароль не будут легитимными или закрывает форму

*Исключение 2 Введенный адрес почты не совпадает с указанным в профиле*

1. Письмо со ссылкой на смену пароля не приходит

*Исключение 3 Введенные пароли не совпадают*

1. Пользователю выводится сообщение, что пароли не совпадают, предлагается ввести пароли снова

2. Пользователь вводит пароль и подтверждает его до тех пор, пока пароли не будут совпадать или закрывает форму

***Сценарий 3 – Управление данными пользователей***

Вариант использования: создание, изменение, удаление данных пользователя

Участники: администратор

Цель: настройка и создание учетных записей, поддержка актуальных данных о пользователях

Краткое описание: администратор вводит данные пользователя, присваивает ему роль, изменяет данные пользователя, удаляет их

Типичный ход событий:

1. Если администратору необходимо создать новую учетную запись, он вводит данные пользователя: ФИО, дата рождения, номер клубной карты (если это клиент или менеджер), логин, пароль, телефон, email, роль, дату создания учетной записи

*Исключение 1 Дублирование аккаунта*

*Исключение 2 Введенные данные некорректны*

1. Если данные пользователя изменились, администратор ищет учетную запись пользователя, меняет данные, сохраняет изменения

*Исключение 1 Дублирование аккаунта*

*Исключение 2 Введенные данные некорректны*

1. В случае ошибочно введенных данных, администратор удаляет учетную запись

Раздел исключений:

*Исключение 1 Дублирование аккаунта*

1. Если введенные данные дублируют уже существующие данные администратору выводится об этом сообщение, запись не добавляется (не изменяется)
2. Администратор повторяет попытку ввода до тех пор, пока данные не станут уникальными или закрывает форму

*Исключение 2 Введенные данные некорректны*

1. Если введенные администратором данные не проходят проверки корректности, система выводит сообщение об этом, запись не создается (не изменяется)
2. Администратор повторяет попытку ввода данных до тех пор, пока данные не станут уникальными или закрывает форму

***Сценарий 4 – Управление данными фитнес-клуба***

Вариант использования: управление данными фитнес-клуба (о самом фитнес-клубе, о зонах, о групповых программах, о графике работы, о клубных картах)

Участники: администратор

Цель: актуальное описание фитнес-клуба, зон клуба, групповых программ, графика работы, клубных карт

Краткое описание: администратор создает, изменяет, удаляет данные фитнес-клуба, зон клуба, групповых программ, графика работы, клубных карт

Типичный ход событий:

1. Администратор вводит данные

*Исключение 1 Введенные данные некорректны*

*Исключение 2 Дублирование данных*

1. В случае если данные претерпели изменения, администратор изменяет данные

*Исключение 1 Введенные данные некорректны*

*Исключение 2 Дублирование данных*

1. В случае если фитнес-клуб или зона клуба закрыты, администратор проставляет дату закрытия

*Исключение 3 Дата закрытия меньше даты открытия*

Раздел исключений:

*Исключение 1* *Введенные данные некорректны*

1. В случае если данные были ошибочно введены администратор их удаляет

Раздел исключений:

*Исключение 1* *Введенные данные некорректны*

1. Если администратор ввел некорректные данные, ему выводится сообщение, после чего данные не сохраняются
2. Администратор вводит данные до тех пор, пока они не станут корректными или закрывает форму

*Исключение 2 Дублирование данных*

1. В случае попытки повторного ввода, администратору выводится сообщение, данные не сохраняются
2. Администратор вводит данные до тех пор, пока не введет уникальные или не закроет форму

*Исключение 3 Дата закрытия меньше даты открытия*

1. В случае если дата закрытия меньше даты открытия, администратору выводится сообщение, данные не сохраняются
2. Администратор вводит данные до тех пор, пока не введет корректные или не закроет форму

***Сценарий 5 – Создание расписания групповой программы***

Вариант использования: поддержка актуальности расписания групповых программ

Участники: менеджер

Цель: создание, удаление, отмена, изменение записи о предстоящем занятии групповой программы

Краткое описание: менеджер создает записи о предстоящем занятии групповой программы, указывает предполагаемое количество занимающихся, тренера, если это необходимо; в случае отмены занятия менеджер помечает занятие как отмененное, сообщает клиентам об отмене занятий (по телефону), также у клиентов появляется сообщение об отмененном занятии; изменяет его описание; если никто не успел записаться, удаляет занятие.

Типичный ход событий:

1. В случае если необходимо создать расписание предстоящего занятия, менеджер создает записи, содержащие данные о самом занятии, о дате и времени, о тренере, о предполагаемом количестве занимающихся для комфортной тренировки.
2. Если занятие отменяется по каким-то причинам, менеджер помечает занятие как отмененное, сообщает об этом клиентам, в планируемых посещениях клиента появляется отметка об отмене занятия
3. Если занятие изменилось или поменялось описание, менеджер изменяет данные о предстоящем занятии
4. Если никто из клиентов не успел записаться, то менеджер удаляет занятие, при необходимости

***Сценарий 6 – Ввод информации о планируемом посещении фитнес-клуба***

Вариант использования: запись данных о планируемом посещении клиента

Участники: клиент, менеджер

Цель: создание записи о планируемом посещении

Краткое описание: клиент, ориентируясь на статистику планируемых посещений клиентов, вводит информацию о планируемом визите фитнес-клуба – выбирает дату, время занятия, зону посещения, доступные по его клубной карте и незакрытые на выбранную дату

Типичный ход событий:

1. Клиент вводит данные о планируемом визите

*Исключение 1 – На указанное время уже есть планируемое посещение*

*Исключение 2 – Фитнес-клуб или зона фитнес-клуба закрыты на выбранную дату*

*Исключение 3 – Клубная карта на выбранную дату не действительна*

1. В сетке расписания увеличивается количество посетителей на единицу

Раздел исключений:

*Исключение 1 – На указанное время уже есть планируемое посещение*

1. Клиент вводит дату и время, фитнес клуб, зону (группу) планируемого занятия, совпадающие с уже запланированными посещениями этого клиента
2. Клиенту выводится сообщение об ошибке, клиент либо отменяет неактуальное посещение, либо меняет данные текущего

*Исключение 2 – Фитнес-клуб или зона фитнес-клуба закрыты на выбранную дату*

*Исключение 3 – Клубная карта на выбранную дату не действительна*

1. Клиенту выводится сообщение, он вводит другую дату или обращается к администратору

***Сценарий 7 – Просмотр своих запланированных посещений на выбранную дату, удаление неактуальных***

Вариант использования: просмотр планируемых посещений на выбранную дату

Участники: клиент, менеджер

Цель: актуализация своих заявок на посещение

Краткое описание: клиент просматривает планируемые посещения

Типичный ход событий:

1. Клиент выбирает дату, которая представляет интерес
2. Клиент просматривает запланированные посещения, удаляет неактуальные

***Сценарий 8 – Просмотр детальной информации о планирующих посещение***

Вариант использования: просмотр ФИО и номеров телефонов клиентов, запланировавших визит

Участники: менеджер

Цель: контроль загруженности фитнес-клуба

Краткое описание: менеджер просматривает планируемые посещения по дате, времени, фитнес-клубу, зоне

Типичный ход событий:

1. Менеджер вводит дату, выбирает фитнес-клуб и зону

*Исключение 2 – Фитнес-клуб или зона фитнес-клуба закрыты на выбранную дату*

*Исключение 3 – Клубная карта на выбранную дату не действительна*

1. Менеджер просматривает ФИО и номера телефонов клиентов, планирующих посещение с выбранными параметрами

Раздел исключений:

*Исключение 2 – Фитнес-клуб или зона фитнес-клуба закрыты на выбранную дату*

*Исключение 3 – Клубная карта на выбранную дату не действительна*

1. Менеджеру выводится сообщение, он вводит другую дату или обращается к администратору

**Модель предметной области**

После определения функциональных требований к приложению и разработки прецедентов требуется разработать **модель предметной области** – определить основные сущности в терминах предметной области.

***Сценарий 1 – Авторизация и аутентификация***

Представлено на рисунке 2.

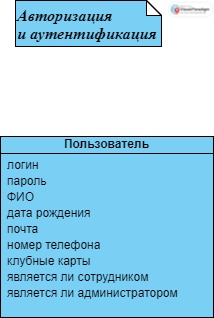


Рисунок 2 – Модель предметной области авторизации и аутентификации

***Сценарий 2 – Ввод пользовательского пароля***

Представлено на рисунке 3.



Рисунок 3 – Модель предметной области ввода пользовательского пароля

***Сценарий 3 – Управление данными пользователей***

Представлено на рисунке 4.



Рисунок 4 – Модель предметной области управления данными пользователей

***Сценарий 4 – Управление данными фитнес-клуба***

Представлено на рисунке 5.

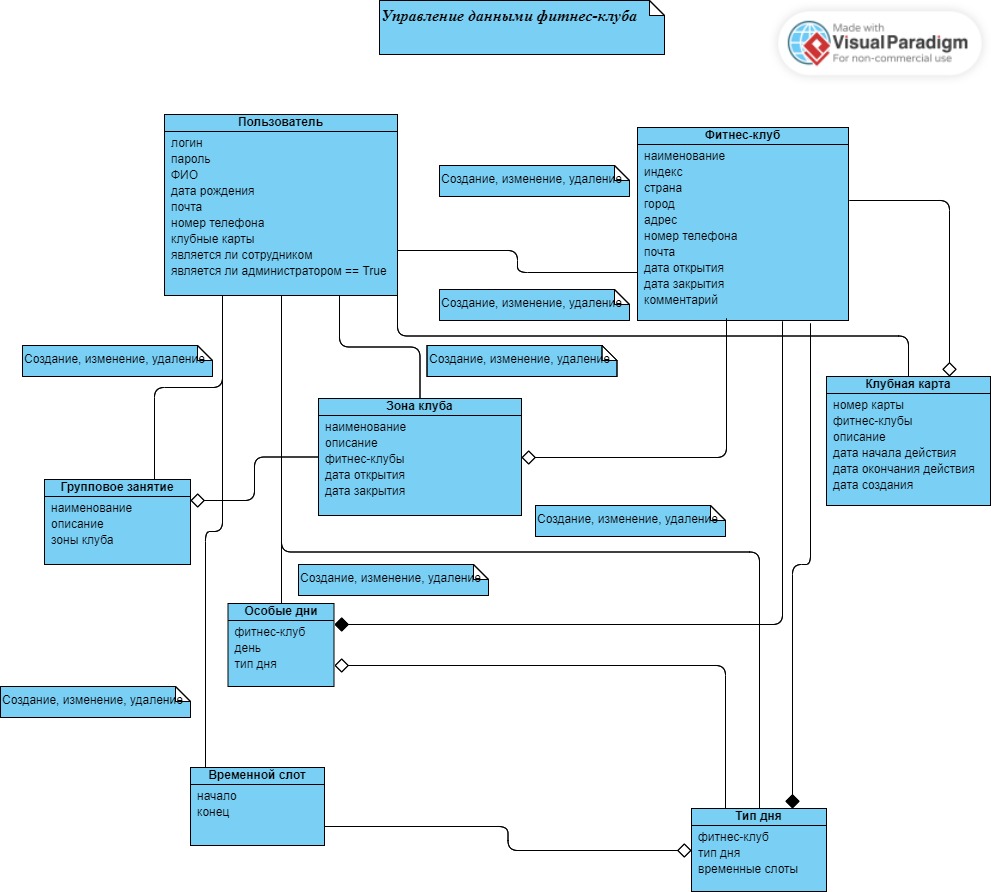


Рисунок 5 – Модель предметной области управления данными фитнес-клуба

***Сценарий 5 – Создание расписания групповой программы***

Представлено на рисунке 6.

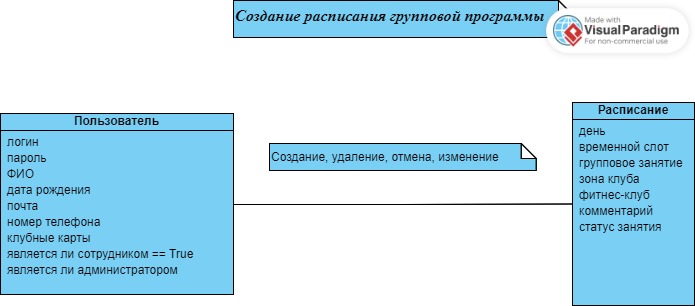


Рисунок 6 – Модель предметной области создания расписания групповой программы

***Сценарий 6 и 7 – Ввод информации о планируемом посещении фитнес-клуба. Просмотр своих запланированных посещений на выбранную дату, удаление неактуальных***

Представлено на рисунке 7.



Рисунок 7 – Модель предметной области ввода информации о планируемом посещении фитнес-клуба, просмотра своих посещений на выбранную дату, удаление неактуальных

***Сценарий 8 – Просмотр детальной информации о планирующих посещение***

Представлено на рисунке 8.

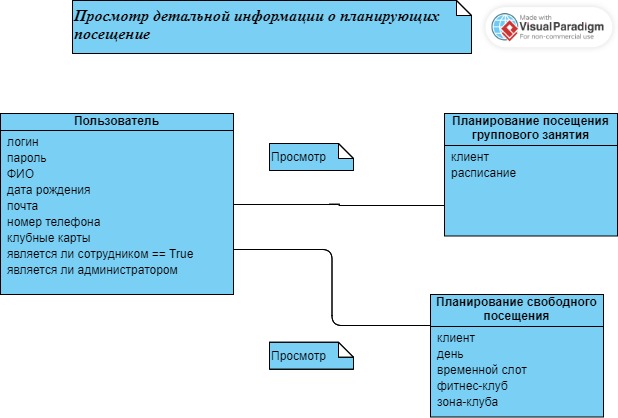


Рисунок 8 – Модель предметной области просмотра детальной информации о планирующих посещение

**Архитектура приложения**

Представлена на рисунке 9.

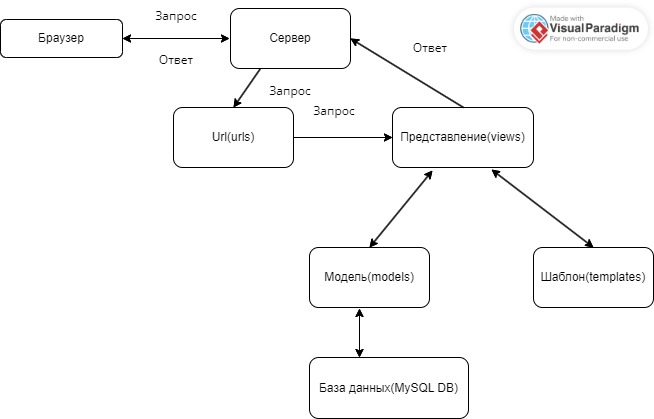
****

Рисунок 9 – Архитектура приложения

**URLs:** хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения (view) в виде аргументов.

**Views:** view (англ. «отображение») — это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.

**Models:** модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

**Templates:** template (англ. «шаблон») — это текстовый файл, определяющий структуру или разметку страницы (например HTML-страницы), с полями для подстановки, которые используются для вывода актуального содержимого. *View* может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (*model).* Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

Клиент с помощью браузера посылает запрос серверу, сервер сопоставляя url-адрес, вызывает один из методов представления, представление получает посредством моделей данные из базы данных (если это необходимо) и выводит в определенный в методе шаблон, далее передает отображение серверу, а тот, в свою очередь, представляет результат в браузере клиенту.

**Диаграмма классов**

Представлена на рисунке 10.

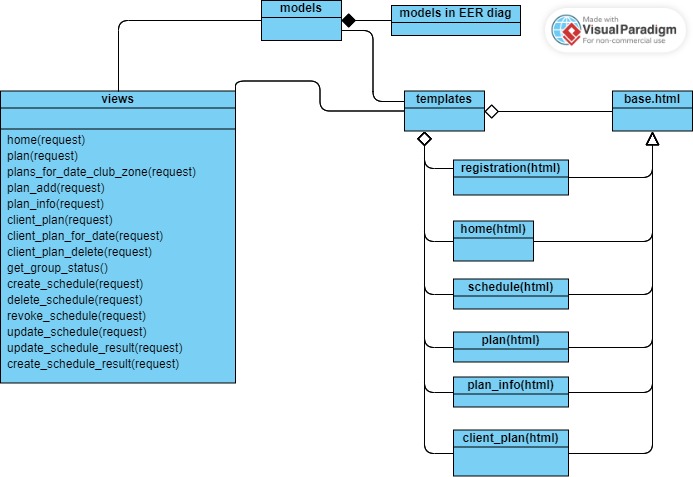


Рисунок 10 – Диаграмма классов

Models представлены на EER-диаграмме в разделе разработки и создании базы данных.

На диаграмме templates представлены в виде групп, так как их очень много. Все они наследуются от шаблона base, в котором создается боковая панель, с навигацией по функциональным блокам программы (стартовая страница – планируемые посещения клиентов, личные планируемые посещения, регистрация в системе).

Подробнее всего, полагаю стоит рассмотреть views и его методы, разработанные мной, не предоставляемые Django.

1. home (request) – вызывает шаблон home.html, передает форму для ввода даты, формирует POST-запрос к методу plans\_for\_date\_club\_zones
2. plans\_for\_date\_club\_zones (request) – посредством POST-запроса получает дату, введенную пользователем, проверяет посредством модели UserAdditionalInfo, есть ли у клиента действующая клубная карта (модель ClubCard) в выбранный день, далее формирует списки доступных клиенту фитнес-клубов (модель FitnessClub) и соответствующих им зон (модель ClubZone), выводит шаблон plans\_for\_date\_club\_zone.html, передает в форму либо сообщение о недействующей клубной карте, либо списки доступных фитнес-клубов и их зон, формирует POST-запрос к методу plan
3. plan (request) - посредством POST-запроса получает выбранную дату, фитнес-клуб и зону, проверяет, что фитнес-клуб (модель FitnessClub) и зона (модель ClubZone) на эту дату функционируют, если фитнес-клуб или зона не работают, выводит сообщение в шаблоне plan.html. Если все в порядке, далее анализируется день: проверяется относится ли он к особым дням – праздничным или предпраздничным (модель SpecialDay), если нет, то будний это день или выходной. Далее посредством модели DayType (для каждого объекта этой модели описаны временные слоты из модели TimeSlot) определяются временные слоты, которые будут выводиться пользователю. После этого анализируется есть ли у выбранной зоны расписание (модель Schedule). Если есть, то посредством модели Schedule ищутся объекты, соответствующие критериям отбора - дата, слот, фитнес-клуб, зона, также посредством модели PlanGroup ищутся заявки на посещение этих групповых программ, ведется их подсчет. Если нет ни одного составленного расписания, то посредством модели Plan ищутся и подсчитываются заявки на свободное посещение. В шаблон plan.html передается дата, признак групповой зоны, временные слоты, описание групповой программы, количество планирующих посещение. Если это групповое занятие и оно не отменено, то в шаблоне появляется кнопка «Запись», с помощью которой формируется POST-запрос на создание заявки plan\_add. Если это свободное посещение, то кнопка «Запись» отображается без условий. Если пользователь является сотрудником, в шаблоне напротив слотов появляется кнопка «Показать», посредством которой можно сформировать POST-запрос на отображение детальной информации plan\_info. В этом же случае напротив слота без расписания появляется кнопка «Создать расписание», посредством которой можно сформировать POST-запрос на создание расписания create\_schedule. Если расписание уже есть, но записаться еще не успели, то появляется «Удалить занятие» (delete\_schedule), если уже успели, то «Отменить занятие» (revoke\_schedule), и «Изменить занятие» (update\_schedule).
4. plan\_add(request) – посредством POST-запроса получает признак группового занятия. Если занятие групповое, передается идентификатор расписания (модель Schedule) и идентификатор пользователя – посредством модели PlanGroup создается новая заявка на посещение группового занятия. Если занятие без группы, то передается идентификатор пользователя, дата, временной слот, фитнес-клуб, зона клуба и посредством модели Plan добавляется запись на свободное посещение. Создание записи происходит только в случае ее уникальности, после чего в шаблон plan\_add.html передается сообщение об успешном добавлении записи. Если запись неуникальна, передается сообщение о том, что клиент уже записан на этот слот.
5. plan\_info(request) – посредством POST-запроса получает данные о дате, являются ли заявки заявками на групповое занятие. Если это групповое занятие, получает идентификатор расписания и посредством модели PlanGroup ищет записавшихся клиентов и с помощью моделей User и UserAdditionalInfo ищет ФИО и телефон клиента, выводит это в шаблон plan\_info.html . Если это свободное посещение, процесс похож, за исключением того, что в POST-запросе передаются зона, фитнес-клуб и временной слот, а поиск происходит с помощью модели Plan.
6. create\_schedule(request) – посредством POST-запроса получает данные о дате, временном слоте, фитнес-клубе, зоне клуба, из шаблона create\_schedule.html – информацию о групповом занятии и статусе занятия. Посредством шаблона create\_schedule.html формируется POST-запрос на create\_schedule\_result куда передаются указанные данные.
7. create\_schedule\_result(request) - посредством POST-запроса получает данные о дате, временном слоте, фитнес-клубе, зоне клуба, групповой занятии, статусе занятия. Создается расписание (модель Schedule), в шаблон create\_schedule\_result.html передается сообщение об успешно созданном расписании, в случае его уникальности, в противном случае – передается сообщение о существовании такого слота.
8. delete\_schedule(request) – посредством POST-запроса получает идентификатор расписания, удаляет расписание,

**Подход к разработке системы**

В связи с тем, что проект небольшой, с отсутствием четкой спецификации, задание и требования становятся более ясными в процессе разработки, полагаю, подход к разработке системы должен быть гибким. В связи с тем, что я разрабатываю систему одна, результат примерно ясен, задачи для меня проще всего разбить на подзадачи, поэтому, вероятно, ближе всего модель Kanban.

**Средства реализации**

Так как мое приложение не предполагает сложной визуализации, не будет использовать особые возможности устройства, на котором будет запускаться, оно не очень сложное само по себе, должно быть максимально доступным для широкого круга пользователей, мною был выбран способ реализации в виде web-приложения.

Приложение будет написано на языке Python. Преимуществом этого языка программирования для моей работы является большая скорость разработки, простой и понятный синтаксис, гибкость, множество фреймворков и библиотек, кроссплатформенность и то, что он подходит в том числе и для web-разработки. Версия выбрана 12, как стабильная и актуальная на момент разработки.

Также в проекте будет использован фреймворк для web-разработки Django 5.0.3. Выбор был сделан в пользу него из-за скорости разработки, приложение собирается как конструктор, есть готовые механизмы авторизации, простые шаблоны веб-страниц, предлагается административный интерфейс для управления контентом, есть возможность поддержки работы с MySQL, легко запустить сервер.

При проектировании базы данных была выбрана реляционная модель, так как модель данной предметной области хорошо в нее укладывается. Реляционные базы данных позволяют обрабатывать сложные запросы к структурированным данным, сохраняя их целостность и согласованность. В качестве средства управления базой данных была выбрана MySQL 8.3, так как она простая, надежная, быстрая и не перегруженная функциями.

**Разработка и создание БД**

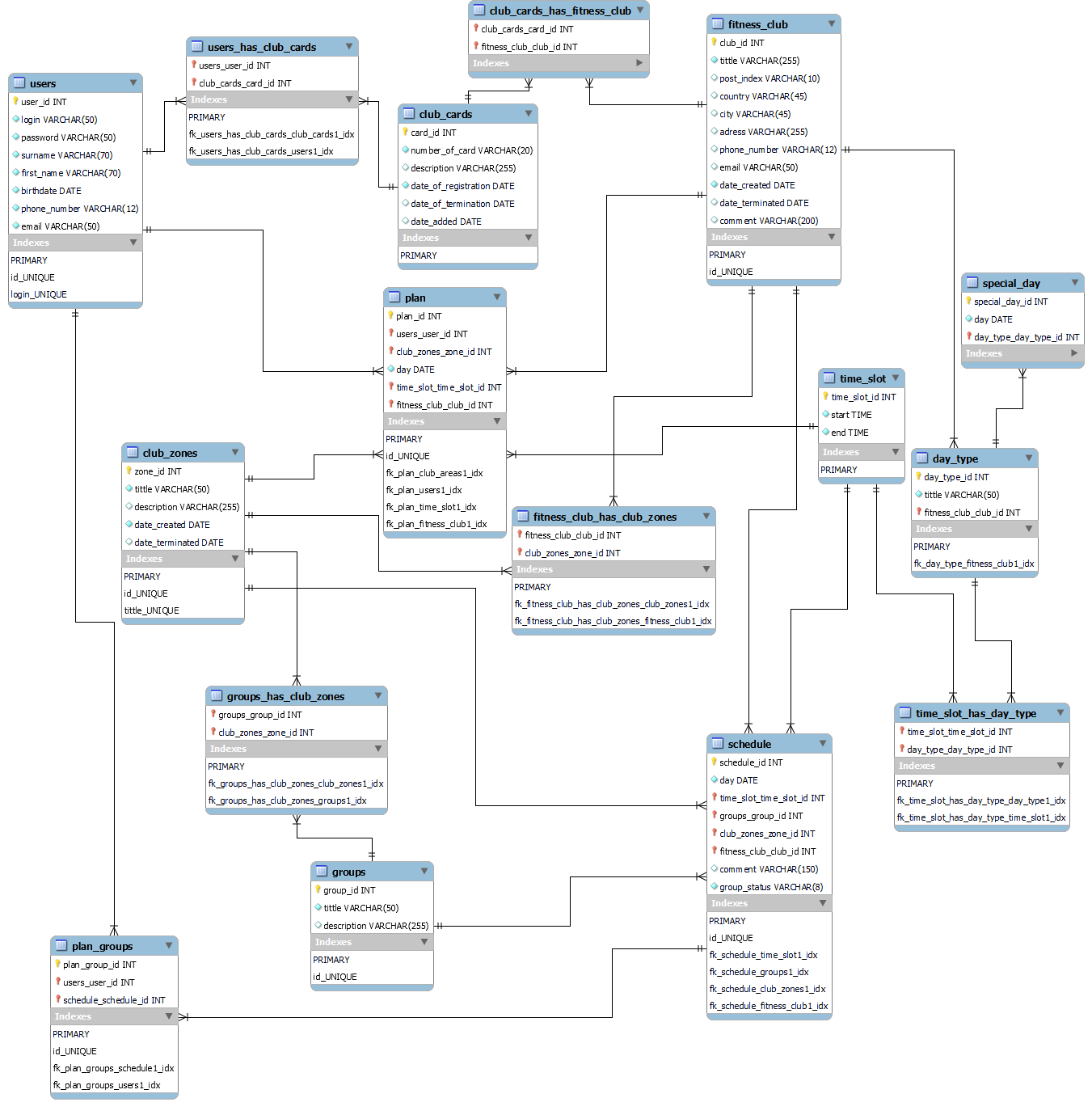
На рисунке 11 представлена EER-диаграмма

Рисунок 11 – EER-диаграмма

Голубым отмечены поля обязательные к заполнению.

Таблицы находятся в первой нормальной форме, так как все их атрибуты атомарны и ключевые поля не являются пустыми.

Так как все поля таблиц, не входящие в первичный ключ, зависят от первичного ключа, можно сделать вывод, что база данных соответствует второй нормальной форме

Ни одно из не ключевых полей таблиц не зависит от любых других не ключевых полей, можно сделать вывод, что база данных находится в третьей нормальной форме.

Так как база данных соответствует третьей нормальной форме, можно сделать вывод, что нормализация данных проведена успешно.

**Разработка серверной части приложения**

Разработку серверной части приложения я начала с создания моделей. Была создана EER-диаграмма в MySQL Workbench, с созданной модели был сгенерирован SQL-код, на основе этого кода была создана база данных. Далее средствами Django я преобразовала существующую базу данных в модели, после чего база данных была удалена и создана пустая (без таблиц) с тем же наименованием. Через settings.py было настроено подключение к базе данных. Поля моделей были откорректированы, прописаны наименования полей при отображении, были описаны внутренние классы Meta, а именно управление классами посредством Django, порядок сортировки по умолчанию, наименование классов в отображении, ограничения – уникальные сочетания полей, значения полей. В настройках был указан язык, часовой пояс.

Далее я настроила через admin.py административный доступ для приложения.

Произвела миграцию с помощью manage.py, создала административный доступ.

Далее основываясь на примере разработки сайта библиотеки я создала свой базовый шаблон добавила в него боковую панель навигации и реализовала возможность авторизации и аутентификации.

В urls.py я прописала путь ссылающийся на домашнюю страницу, а во views.py метод home посредством которого открывался шаблон home.html (наследовавший базовый шаблон).

Далее я создала форму для ввода даты, посредством которой я получала введенную дату пользователем в отображениях.

После этого я создала отображения для планируемых посещений клиента, для планируемых посещений клиентов, для создания заявок на посещение, для работы с расписанием, в urls.py прописала пути, а во views.py функции для вывода отображений.

После в settings.py я прописала параметры для отправки сообщения по почте для сброса пароля.

Написала документацию.

**Способ развертывания системы**

Так как мой проект является учебным, он небольшой, с небольшой нагрузкой, то самым оптимальным выбором для развертывания системы считаю виртуальный хостинг. Из всего многообразия сервисов выбрала pythonanywhere.com , так как он позволяет вести разработку прямо в браузере, подходит для быстрой разработки несложных приложений, поддерживает Django, MySQL, в нем легко работать с GitHub.

Для развертывания системы я зарегистрировалась на pythonanywhere.com, скопировала свой проект с GitHub, создала виртуальную среду, установила Django и MySQL, настроила свое web-приложение, создала пользователя и базу данных, настроила подключение своего приложения к базе данных, запустила миграцию для создания таблиц в базе данных по моделям Django, создала административный доступ, настроила использование статических файлов, выключила режим отладки, перезагрузила сервер.

Для администратора фитнес-клуба я должна передать учетные данные административного доступа и ссылку на ресурс.

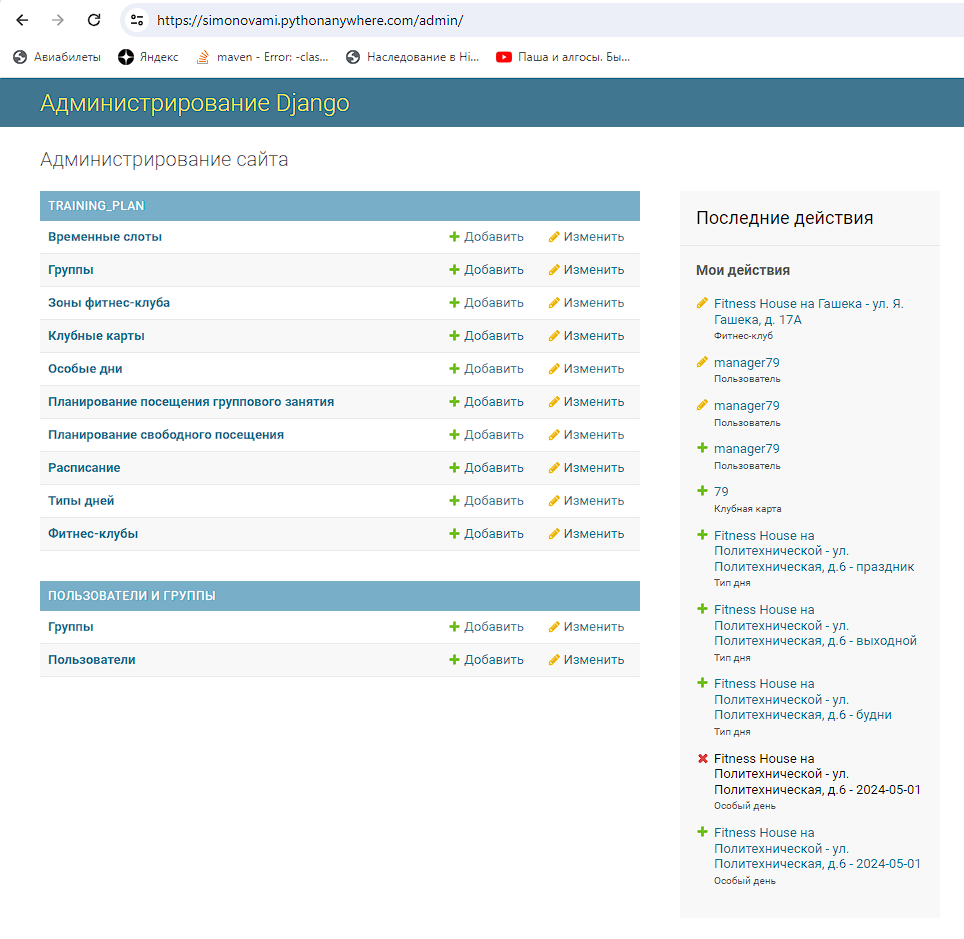
**Тестирование**

Тестирование системы было произведено одновременно с четырех устройств. Способ тестирования был выбран ручной, в связи с ограниченным временем на реализацию проекта. Были проверены все сценарии из вариантов использования. Тестирование считаю пройденным успешно.

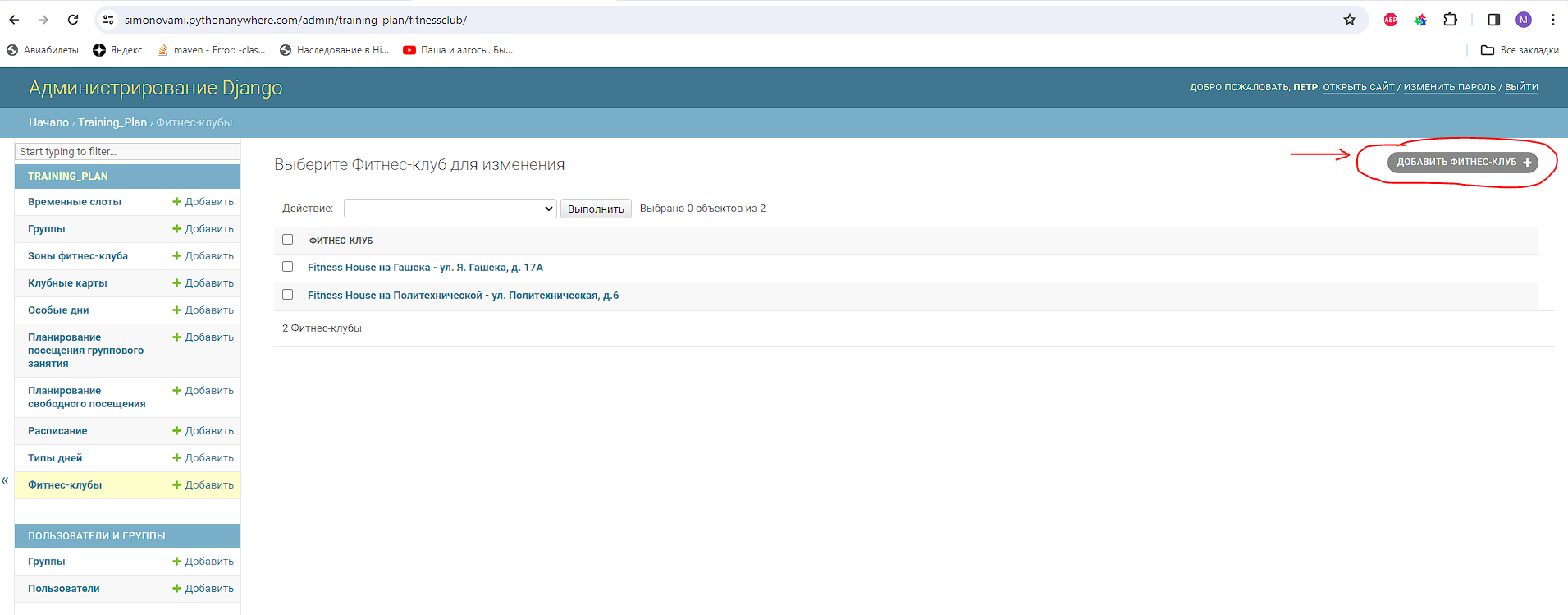
**Инструкции**

Для администратора

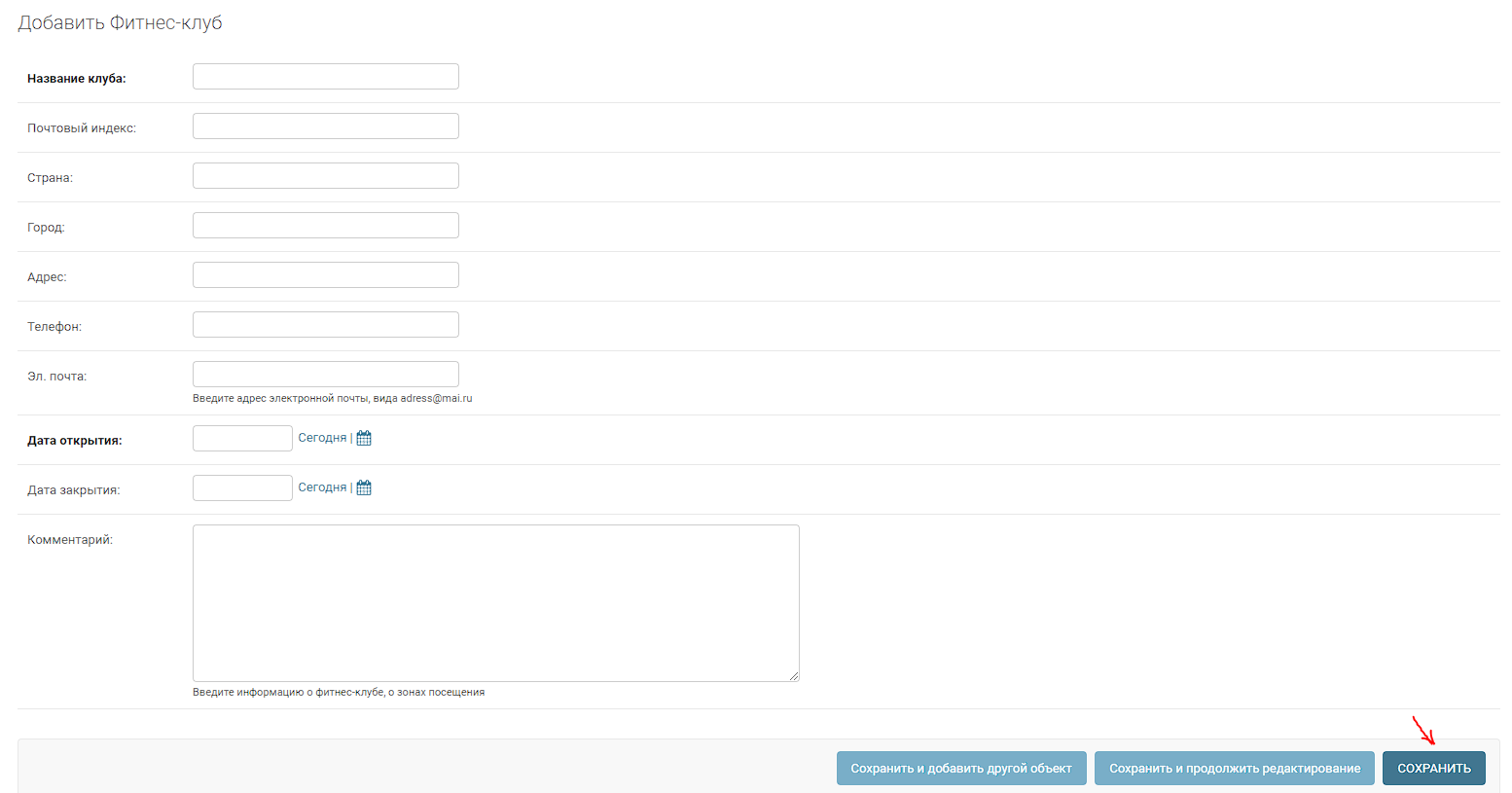
1. Необходимо получить у разработчика учетные данные административного доступа
2. После этого перейти по ссылке <https://simonovami.pythonanywhere.com/admin/>
3. Ввести полученные у разработчика логин и пароль административного доступа. Вы попадете в раздел «Администрирование Django», представленного на рисунке 12.

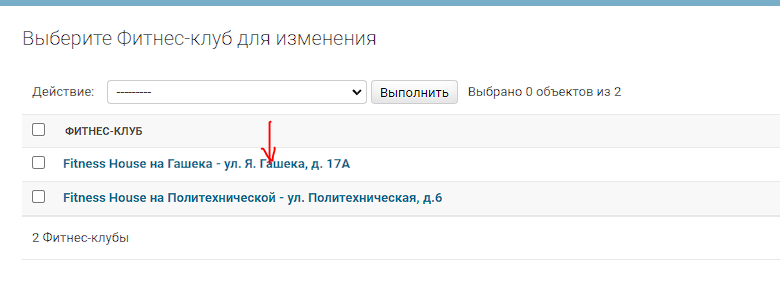


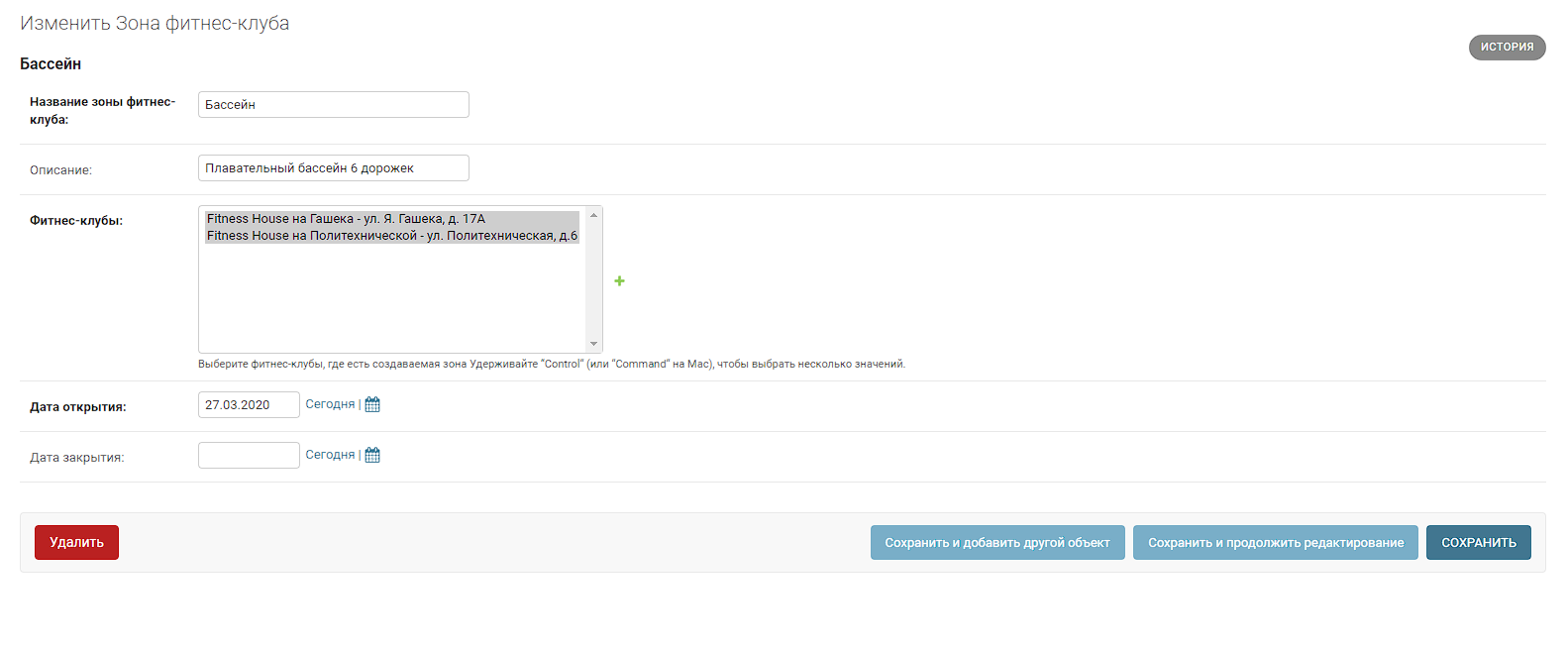
1. Для начала работы необходимо создать фитнес-клуб. Для этого перейдите в раздел фитнес-клубы. На открывшейся странице нажмите на кнопку «Добавить фитнес-клуб». Представлено на рисунке.



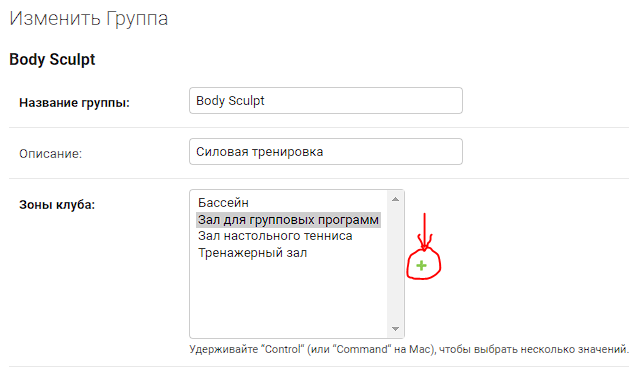
1. Введите данные фитнес-клуба в форме «Добавить фитнес-клуб», нажмите «Сохранить». Обязательные к заполнению поля выделены жирным шрифтом. Представлено на рисунке.



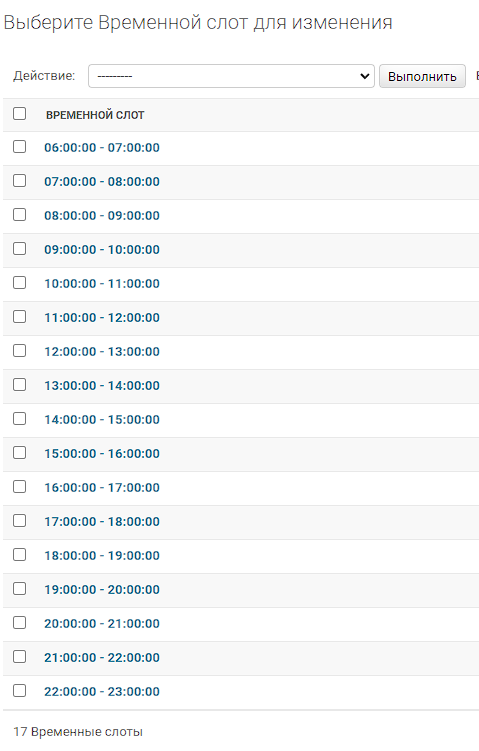
1. Если необходимо изменить данные клуба перейдите в раздел «Выберите фитнес-клуб для изменения», выберите необходимый клуб, щелкнув по его наименованию мышью. Представлено на рисунке . В случае если клуб закрылся, можно поставить дату закрытия, она должна быть больше даты открытия. Если необходимо, чтобы клиентам выводилось предупреждение об особенностях функционирования клуба, заполните раздел «Комментарий».
2. После этого необходимо создать зоны фитнес-клуба, процесс похож на добавление фитнес-клуба, за исключением того, что одно и тоже наименование зоны может быть в разных фитнес-клубах. В разделе фитнес-клубы необходимо выбрать все клубы, где предполагается выбранная зоны. Сделать это можно зажав «Ctrl» и щелкнув на несколько позиций. Представлено на рисунке .



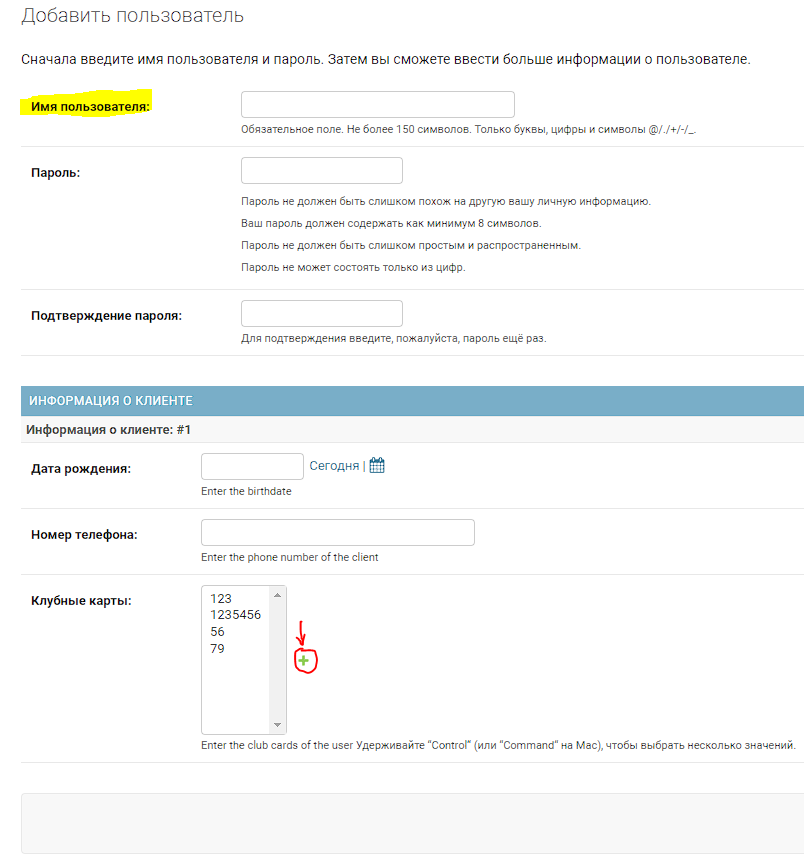
1. После этого нужно создать группы. Необходимо отметить, что для групповых занятий должна быть выделена своя определенная зона. Не должно быть зон с групповыми занятиями и со свободными посещениями. Зона должна быть либо для групповых занятий, либо для свободного посещения. Создать зону клуба под групповое занятие можно, нажав на знак плюса в области выбора зон клуба. Представлено на рисунке .



1. Далее необходимо настроить график работы фитнес-клуба. Для этого перейдите в раздел «Временные слоты». Создайте временные слоты, например, следующим образом, представленным на рисунке .



1. После этого необходимо настроить «Типы дней». Типы дней могут быть буднями, выходными, праздниками, препраздничными днями. В случае если в праздники клуб не работает временные слоты можно не указывать вовсе. Расписание будних и выходных дней подтягивается по календарю, праздники и предпраздничные дни необходимо настраивать индивидуально, с помощью раздела «Особые дни». В указанном разделе должны быть описаны все те дни, в которых график работы клуба не совпадает с описанным для будней и выходных.
2. После настройки фитнес-клуба можно переходить к настройке пользователей. Выберите раздел «Пользователи», нажмите на кнопку «Добавить пользователя». «Имя пользователя» является его логином. Требования к паролю: пароль не должен быть похож на личную информацию, должен содержать как минимум 8 символов, не должен быть слишком простым и распространенным, не может состоять только из цифр. Для корректной работы в системе у каждого пользователя должна быть прописана клубная карта. Создать клубную карту можно из настроек пользователя, что является более удобным вариантом добавления клубной карты, для этого нажмите на знак плюсика в разделе клубные карты. Представлено на рисунке .

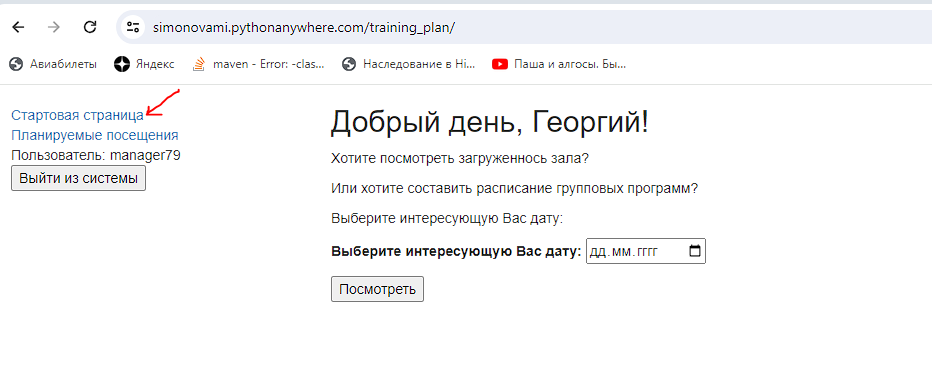


После сохранения пользователя Вы перейдете в раздел «Изменить пользователя», где можно будет заполнить персональную информацию (обязательно прописывайте почту – это необходимо для смены пароля), настроить права доступа. Если Вы настраиваете менеджера, то необходимо отметить галочкой в правах доступа, что пользователь активный и имеет статус персонала, если администратора – активный, статус персонала, статус суперпользователя (тогда будет доступна страница администрирования), если это клиент – активный.

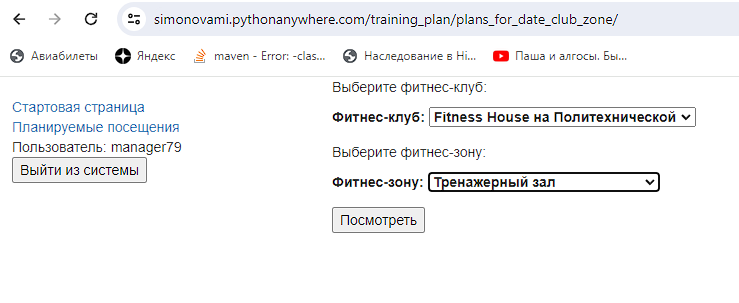
1. Далее чтобы зона клуба стала зоной для групповых занятий, ее нужно инициализировать созданием расписания, которое после создания корректного расписания сможет удалить менеджер.

Для менеджера

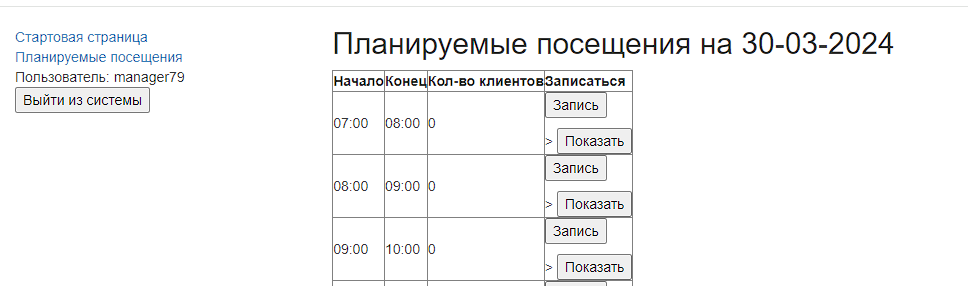
1. Перейдите по ссылке <https://simonovami.pythonanywhere.com/accounts/login/?next=/training_plan/> .
2. Введите логин и пароль пользователя, предоставленный Вам администратором, нажмите «Вход».
3. В случае, если пароль необходимо сменить или Вы его забыли, введите логин, нажмите «Забыли пароль?».
4. Введите почту, нажмите «Сброс пароля».
5. Проверьте почту, должна прийти ссылка для сброса пароля. Если на почту не приходит ссылка для сброса пароля, проверьте спам, обратитесь к администратору с просьбой проверить почту на своем профиле.
6. Перейдите по ссылке, введите пароль заново, зарегистрируйтесь.
7. На панели закладок Вы видите «Стартовую страницу» (которая открывается по умолчанию, в этом разделе можно управлять расписанием, а также создавать заявку на посещение), «Планируемые посещения» (которая отображает заявки пользователя, с возможностью их удаления, и предложение выйти из системы). Представлено на рисунке .



1. На «Стартовой странице» необходимо выбрать интересующую дату, щелкнув на календарик, нажать «Посмотреть».
2. Далее Вам будет предложено выбрать фитнес-клуб и фитнес-зону, согласно Вашей клубной карте, если чего-то не хватает, обратитесь к администратору. Сделайте выбор и нажмите кнопку «Посмотреть». Представлено на рисунке .

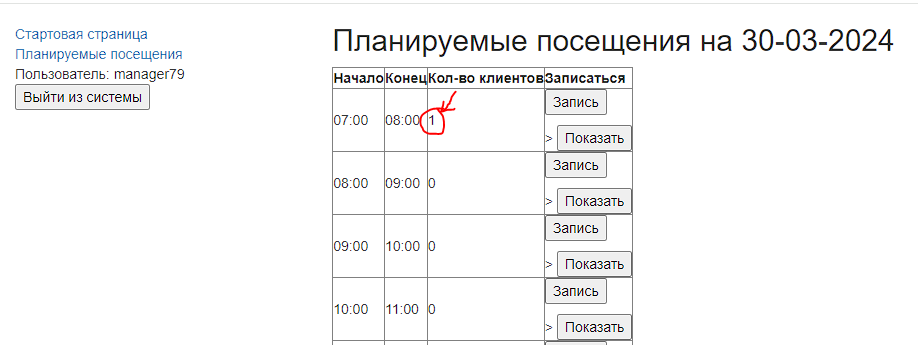


1. В случае, если Вы выбрали зону клуба без групповых занятий, Вам отобразиться форма представленная на рисунке или выведется предупреждение о невозможности просмотра и записи на эту дату.

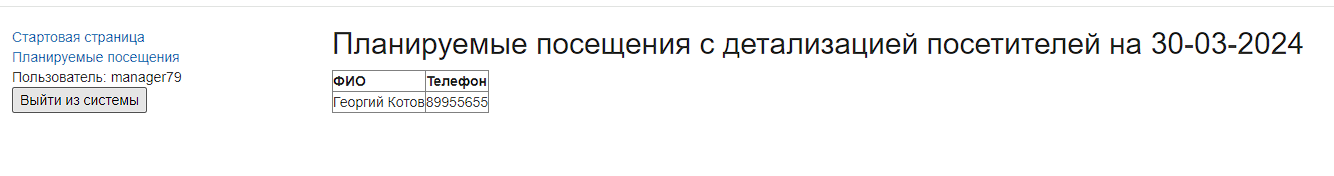


1. Вы можете подать заявку на посещение, нажав на кнопку «Запись» напротив выбранного временного слота, если запись не дублируется, система выдаст сообщение, что запись произведена успешно, если обновить страницу или перейти на «Стартовую» и ввести те же параметры, количество клиентов увеличится на один.

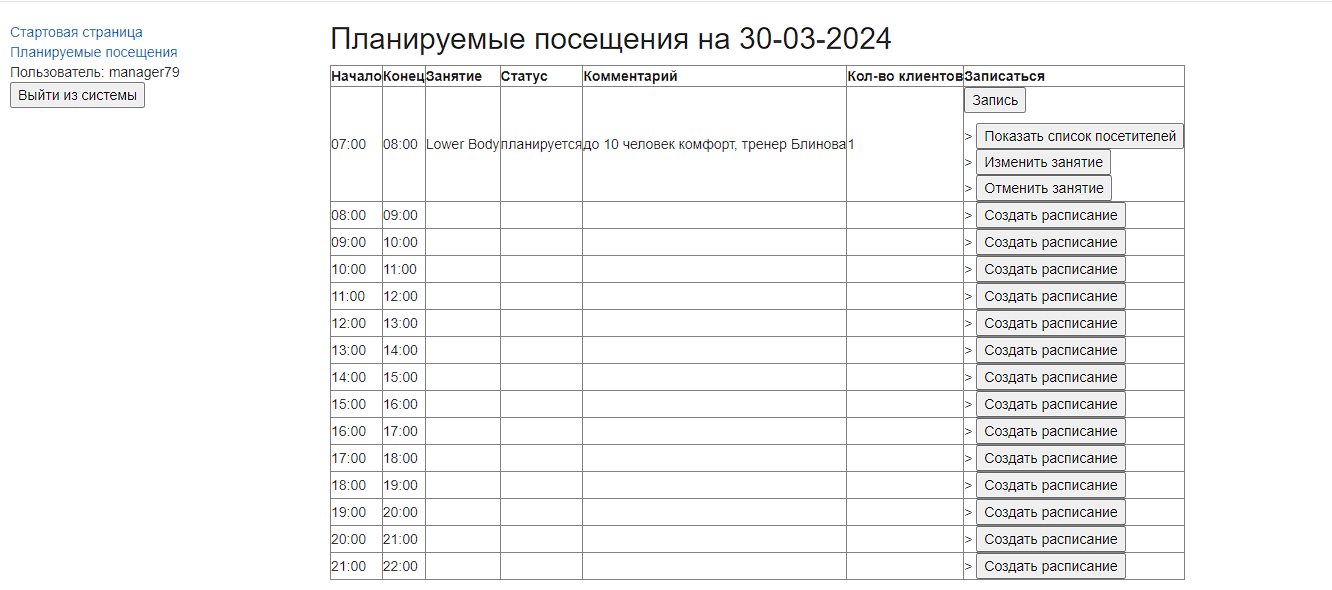
Представлено на рисунке .



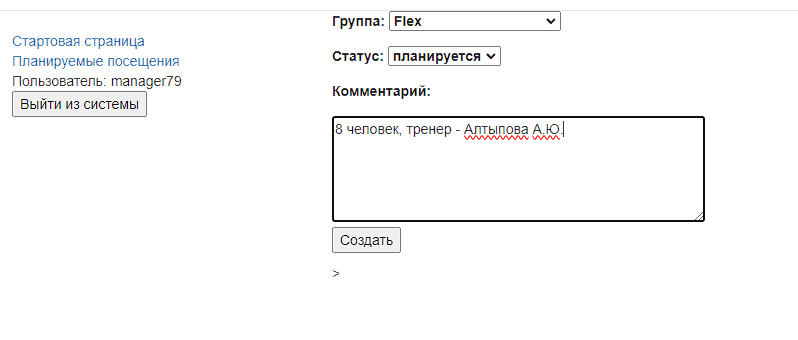
1. В случае если Вам необходимо посмотреть детальную информацию о записанных, можно нажать кнопку «Показать». Представлено на рисунке .



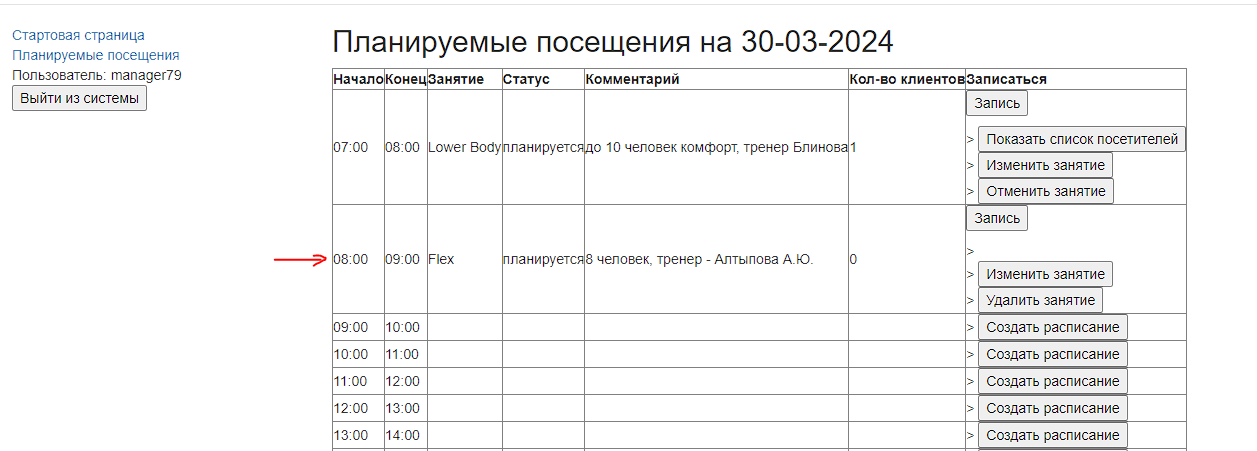
1. Если была выбрана зона для групповых занятий, то форма будет выглядеть как представлено на рисунке .



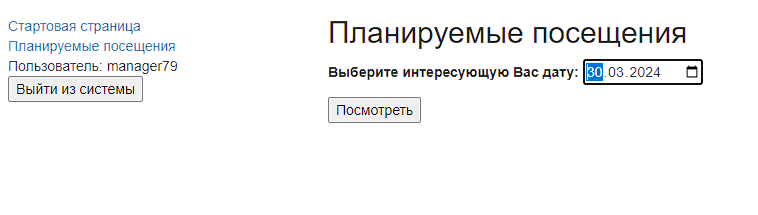
1. Если на слоте нет расписания, то Вам будет предложено его добавить, нажмите на кнопку «Создать расписание», выведется форма создания расписания, как на рисунке . Создайте расписание и нажмите кнопку «Создать».



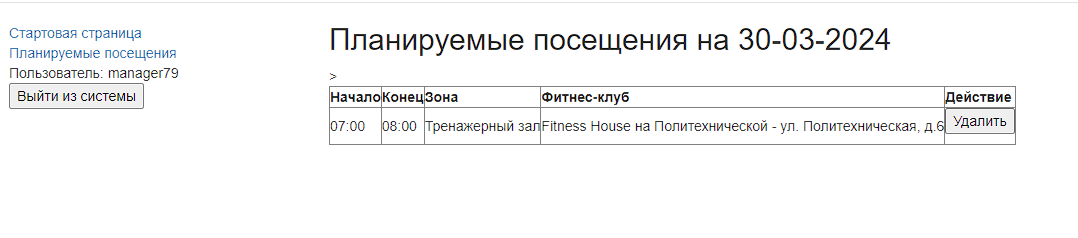
1. После этого в таблице отобразиться созданное расписание. Представлено на рисунке .



1. Если никто из клиентов еще не успел записаться будет доступна кнопку «Удалить занятие».
2. «Изменить занятие» можно в любом случае, вне зависимости от записавшихся.
3. Если клиенты уже успели записаться, то будет возможность «Отменить занятие».
4. В случае если кто-то записался, будет возможность просмотреть детальную информацию о записавшихся, с помощью кнопки «Показать список посетителей».
5. Если занятие не отменено и расписание создано, будет возможность подать заявку о планируемом посещении через кнопку «Запись».
6. В разделе «Планируемые посещения» можно посмотреть свои заявки на выбранную дату. Нажмите «Планируемые посещения», выберите дату, нажмите «Посмотреть». Представлено на рисунке .

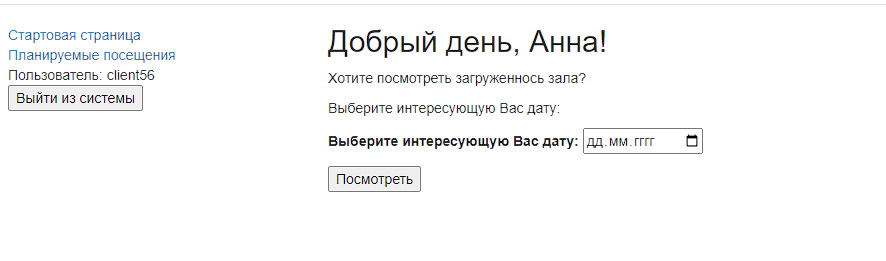


1. Будет выведена таблица с заявками, в случае изменения планов, заявки можно удалять, нажав кнопку «Удалить».

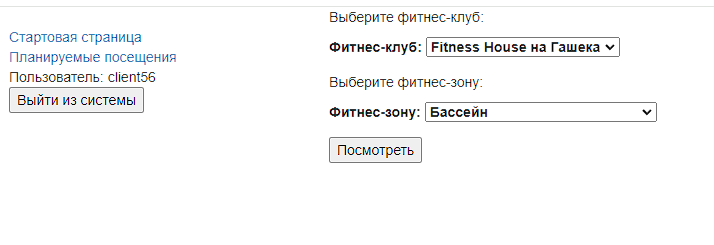


Для клиента

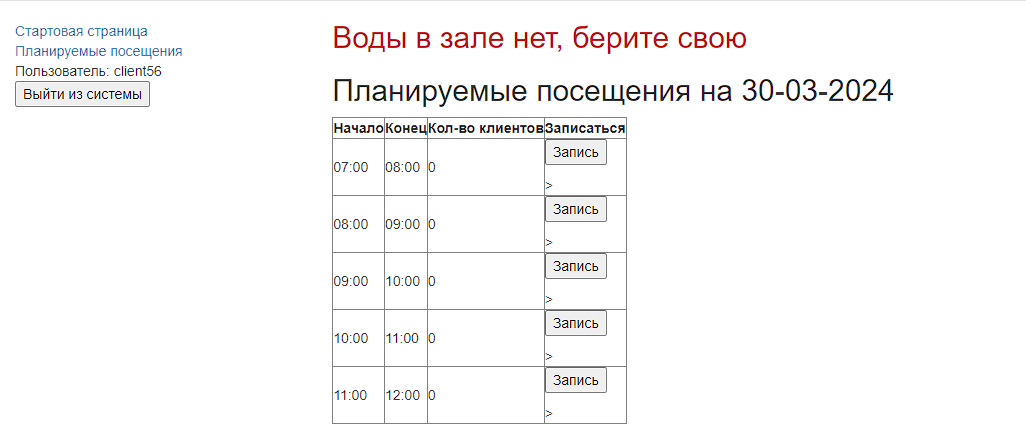
1. Перейдите по ссылке <https://simonovami.pythonanywhere.com/accounts/login/?next=/training_plan/>.
2. Введите логин и пароль пользователя, предоставленный Вам администратором, нажмите «Вход».
3. В случае, если пароль необходимо сменить или Вы его забыли, введите логин, нажмите «Забыли пароль?».
4. Введите почту, нажмите «Сброс пароля».
5. Проверьте почту, должна прийти ссылка для сброса пароля. Если на почту не приходит ссылка для сброса пароля, проверьте спам, обратитесь к администратору с просьбой проверить почту указанную в Вашем профиле.
6. Перейдите по ссылке, введите пароль заново, зарегистрируйтесь.
7. На панели закладок Вы видите «Стартовую страницу» (которая открывается по умолчанию, в этом разделе можно создавать заявку на посещение), «Планируемые посещения» (которая отображает заявки пользователя, с возможностью их удаления, и предложение выйти из системы). Представлено на рисунке .



1. На «Стартовой странице» необходимо выбрать интересующую дату, щелкнув на календарик, нажать «Посмотреть».
2. Далее Вам будет предложено выбрать фитнес-клуб и фитнес-зону, согласно Вашей клубной карте, если чего-то не хватает, обратитесь к администратору. Сделайте выбор и нажмите кнопку «Посмотреть». Представлено на рисунке .

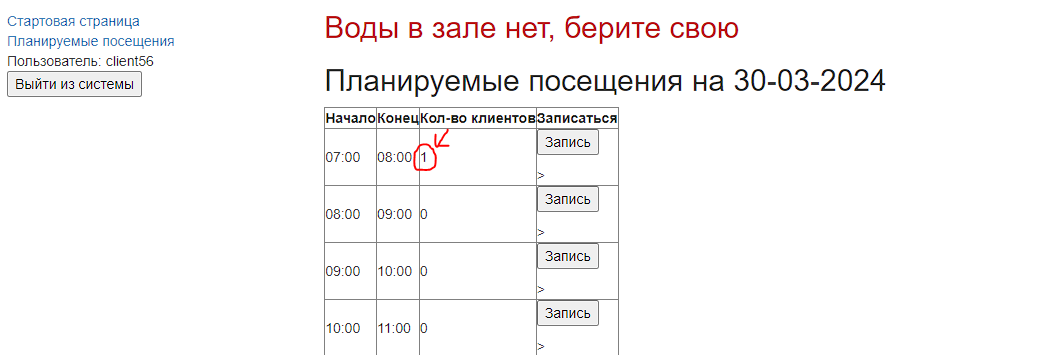


1. В случае, если Вы выбрали зону клуба без групповых занятий, Вам отобразиться форма представленная на рисунке или выведется предупреждение о невозможности просмотра и записи на эту дату.

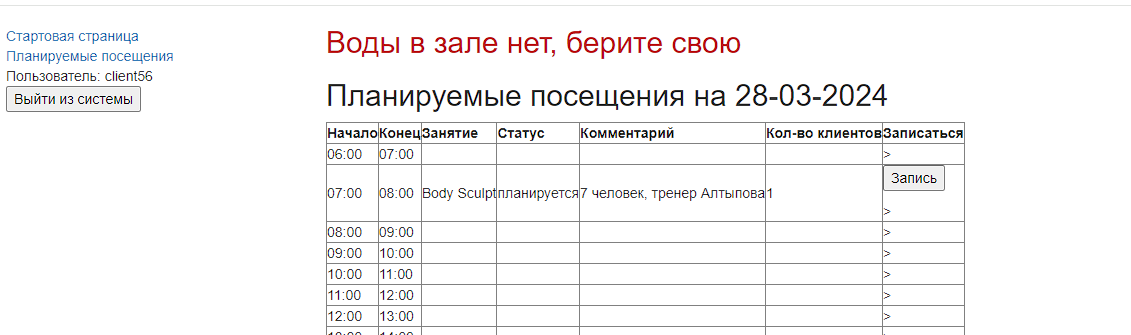


1. Вы можете подать заявку на посещение, нажав на кнопку «Запись» напротив выбранного временного слота, если запись не дублируется, система выдаст сообщение, что запись произведена успешно, если обновить страницу или перейти на «Стартовую» и ввести те же параметры, количество клиентов увеличится на один.

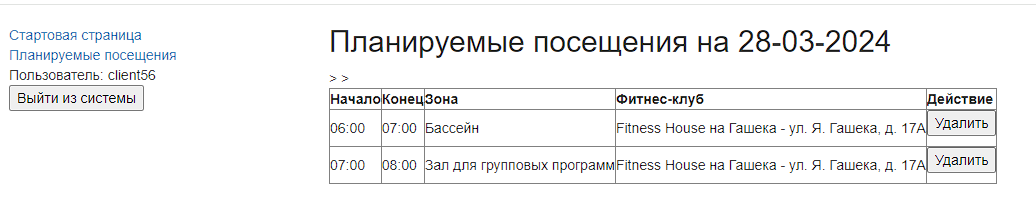
Представлено на рисунке .



1. Если была выбрана зона для групповых занятий, то форма будет выглядеть как представлено на рисунке .



1. Если занятие не отменено и расписание создано, будет возможность подать заявку о планируемом посещении через кнопку «Запись».
2. В разделе «Планируемые посещения» можно посмотреть свои заявки на выбранную дату. Нажмите «Планируемые посещения», выберите дату, нажмите «Посмотреть». Представлено на рисунке .



1. Будет выведена таблица с заявками, в случае изменения планов, их можно удалять, нажав кнопку «Удалить».

**ЗАКЛЮЧЕНИЕ**

В процессе работы было создано клиент-серверное приложение «Training plan», которое является программной системой для контроля загруженности сети фитнес-клубов (фитнес-клуба) и планирования клиентами времени посещения, а также разработана база данных.

В процессе работы: формализованы функциональные требования, проанализирована предметная область, спроектирована диаграмма последовательностей, выбран подход к разработке системы, спроектирована диаграмма классов, выбраны средства реализации, разработана и создана база данных, разработано приложение, разработан способ развертывания системы, проведено тестирование, разработаны инструкции. В связи с этим, считаю, что задачи дипломного проектирования решены.

Система позволяет пользователю, оценивая загруженность фитнес-клуба, планировать свое время посещения, а также позволяет работникам фитнес-клуба контролировать загруженность. В связи с этим, полагаю, что цель дипломной работы достигнута.

В процессе проектирования столкнулась с тем, что при размещении на хостинге поддерживаемая версия Python была 10, а разработка велась на 12, MySQL и Django также можно было использовать более старой версии. Проблема была решена настройкой виртуального окружения в соответствии с возможностями хостинга, а также исключением из моделей описания исключений (доступно в Django, начиная с 5 версии).

Планы усовершенствования системы: спроектировать более современный интерфейс программы, уменьшить число необязательных, повторяющихся действий при работе с системой, а также создание более детальной системы логирования для администратора.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Приложение к решению Ученого совета Университета ИТМО от «29» ноября 2022 г. № 15 «ТРЕБОВАНИЯ К ВЫПУСКНЫМ КВАЛИФИКАЦИОННЫМ РАБОТАМ». – URL: https://student.itmo.ru/files/1314 (дата обращения 16.03.2023).