

Geekbrains

Специальность: **Разработчик**

Тема: Разработка CRM системы для репетитора

Мануковский Сергей Сергеевич

Воронеж, 2023

Оглавление

| | |
|--|----|
| Введение | 3 |
| Глава 1 Введение в CRM..... | 6 |
| 1.1 Обзор понятия CRM (Customer Relationship Management)..... | 6 |
| 1.2 История и эволюция CRM в бизнесе | 8 |
| 1.3 Роль CRM в управлении клиентским опытом..... | 10 |
| 1.4 Применение CRM в образовательной сфере | 12 |
| Глава 2 Технологии и инструменты CRM..... | 14 |
| 2.1 Обзор основных технологий и инструментов в CRM-системах | 14 |
| 2.2. Сравнительный анализ различных CRM-платформ | 15 |
| 2.3 Особенности CRM для репетиторов | 16 |
| Глава 3 Разработка CRM системы для репетитора «ТИЧА» | 17 |
| 3.1 Технологический стек и разработка CRM системы..... | 17 |
| 3.2 Дальнейшее развитие CRM системы ТИЧА: Инновационные шаги в образовании..... | 74 |
| Заключение..... | 76 |
| Список используемой литературы..... | 77 |

Введение

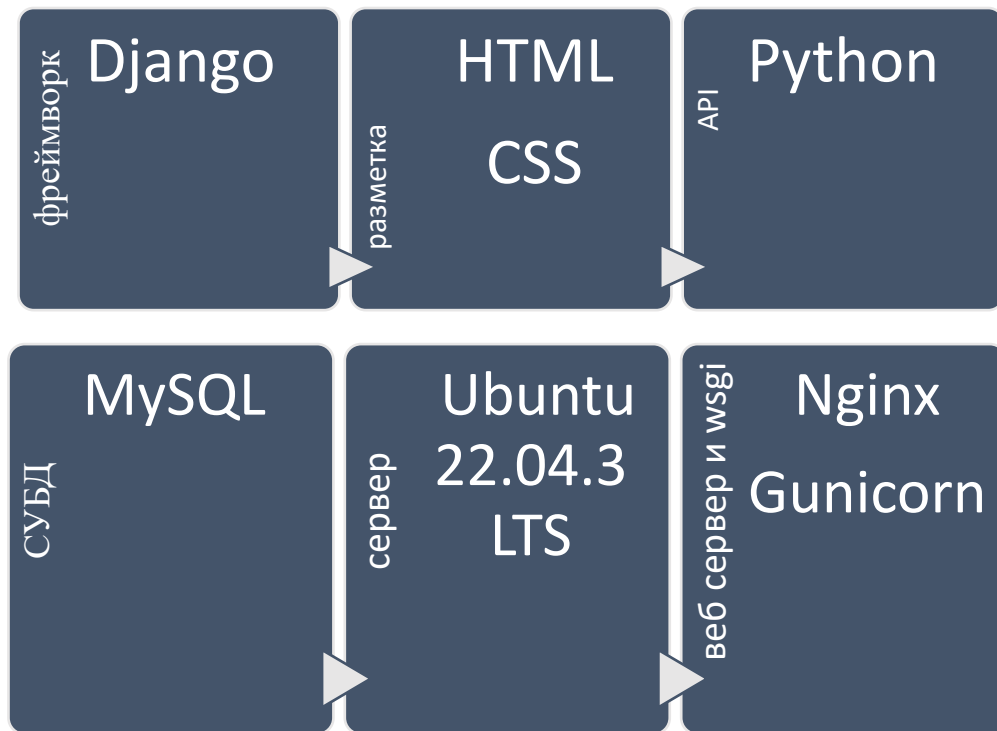
Современные технологии становятся неотъемлемой частью различных областей деятельности, в том числе и образования. В связи с этим дипломное исследование посвящено созданию CRM-системы для репетитора под названием "Тича". Такой проект представляет собой попытку ответить на вызовы современного рынка образовательных услуг и повысить эффективность взаимодействия репетиторов с их учениками. **Предмет проекта:** объектом исследования является разработка и внедрение CRM-системы, специально ориентированной на потребности репетиторов. Такой инструмент может стать ключевым фактором в управлении клиентской базой и оптимизации процесса обучения. **Обоснование темы проекта:** с увеличением конкуренции в сфере образования, эффективное управление клиентами и процессами становится стратегически важным. Разработка CRM-системы для репетиторов представляется актуальной и необходимой для повышения качества предоставляемых услуг. **Цель проекта:** основной целью данного дипломного проекта является создание и внедрение интегрированной CRM-системы "Тича", которая спроектирована для оптимизации взаимодействия репетиторов и их учеников, а также для улучшения процессов управления.

План работы: планируется проведение анализа требований репетиторов, разработка функциональности системы, тестирование и ее последующее внедрение. Этот план позволит охватить все этапы создания и использования CRM-системы.

Решаемая проблема: дипломный проект направлен на решение проблемы неэффективного управления клиентами и не структурированности обучающих процессов у репетиторов. Специализация: Backend и Frontend разработчик

Полезный опыт: в ходе выполнения проекта участник обладает опытом разработки программных продуктов, что включает в себя работу с базами данных и интеграцию систем.

Инструменты и Технологии Проекта



В процессе разработки и внедрения CRM-системы "Тича" будут использоваться современные инструменты и технологии, обеспечивающие эффективность, надежность и безопасность системы:

1. HTML и CSS:

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) используются для разработки веб-интерфейса, обеспечивая структурирование и стилизацию контента.

2. Django:

Django, фреймворк на языке программирования Python, будет использован для быстрой и эффективной разработки веб-приложения. Django предоставляет множество инструментов для работы с базами данных, управления сессиями, обработки форм и многое другое.

3. Python:

Язык программирования Python будет использован для написания бизнес-логики и взаимодействия с базой данных. Python предоставляет удобный синтаксис и широкий набор библиотек для разработки.

4. MySQL:

Система управления базами данных MySQL будет использоваться для хранения и эффективного управления данными приложения.

5. Linux:

Операционная система Linux, как надежная и масштабируемая платформа, обеспечит стабильную работу серверной части приложения.

6. Gunicorn:

Gunicorn (Green Unicorn) будет использован в качестве WSGI-сервера для обработки веб-запросов, обеспечивая высокую производительность и устойчивость.

7. Nginx:

Веб-сервер Nginx будет использован для обеспечения балансировки нагрузки, проксирования запросов и обеспечения безопасности веб-приложения.

Этот стек технологий обеспечит не только надежное функционирование CRM-системы "Тича", но и обеспечит высокую производительность, удобство использования и безопасность данных пользователей. Состав Команды:

Работа над проектом осуществляется индивидуально, что позволяет одному человеку заниматься всеми аспектами разработки и внедрения системы.

Глава 1 Введение в CRM

1.1 Обзор понятия CRM (Customer Relationship Management)

Customer Relationship Management (CRM) представляет собой стратегический подход к управлению отношениями с клиентами, нацеленный на максимизацию их удовлетворенности и укрепление долгосрочных взаимоотношений. В основе концепции CRM лежит идея, что удовлетворенные клиенты, ощущающие индивидуализированный и заботливый подход, более склонны стать лояльными и частыми потребителями.

Основные принципы CRM:

1. Систематизация данных:

CRM начинается с сбора и систематизации данных о клиентах. Это включает в себя базовую информацию (контактные данные, предпочтения) и данные о взаимодействии с продуктами или услугами.

2. Анализ данных:

Проанализированные данные позволяют понять потребности и предпочтения клиентов. Аналитика в CRM выходит за рамки обычного мониторинга транзакций, включая анализ поведенческих шаблонов и тенденций.

3. Персонализированный подход:

Одной из ключевых черт CRM является создание персонализированных взаимодействий с клиентами. Отправка персональных предложений, поддержка индивидуальных запросов и создание уникального клиентского опыта становятся приоритетом.

4. Автоматизация процессов:

Использование CRM также включает в себя автоматизацию ряда процессов, начиная от управления задачами и заканчивая анализом данных. Это способствует повышению эффективности и снижению риска человеческих ошибок.

Цели и задачи применения CRM:

1. Увеличение лояльности клиентов:

CRM направлен на формирование положительного опыта клиентов, что в свою очередь повышает их лояльность к бренду.

2. Повышение эффективности продаж:

Анализ данных позволяет определить наиболее успешные стратегии продаж и сосредоточить усилия на наиболее перспективных клиентах.

3. Улучшение обслуживания:

Персонализированный подход и автоматизация процессов помогают предоставить более высококачественное обслуживание.

4. Развитие долгосрочных отношений:

Целью CRM является не только привлечение новых клиентов, но и укрепление отношений с текущими, способствуя их долгосрочному сотрудничеству.

Общий обзор понятия CRM дает понимание его основных принципов и значимости в современной бизнес-среде. В следующих разделах рассматриваются история, роль CRM в управлении клиентским опытом, а также технологии и инновации в этой области.

1.2 История и эволюция CRM в бизнесе

1.2.1 Предпосылки и ранние формы CRM (до 1990-х)

История Customer Relationship Management (CRM) насчитывает свои корни в ранних попытках компаний устанавливать и поддерживать отношения с клиентами. В начале 20-го века организации использовали карточки клиентов и базы данных для отслеживания основной информации.

1.2.2 Появление Первых CRM-Систем (1980-1990-е)

Переход в компьютерный век привел к созданию первых программных решений для управления отношениями с клиентами. В это время появились ранние CRM-системы, фокусирувавшиеся в основном на автоматизации процессов продаж и управлении контактами.

1.2.3 Развитие концепции CRM (1990-е)

В 1990-е годы концепция CRM стала более комплексной и включала в себя аспекты маркетинга и обслуживания клиентов. Это период характеризовался ростом осведомленности о важности управления отношениями с клиентами для обеспечения конкурентоспособности.

1.2.4 Переход к интегрированным системам (2000-е)

В начале 21 века наблюдался переход к интегрированным CRM-системам, объединяющим данные и процессы в различных областях бизнеса. Этот этап эволюции позволил компаниям лучше понимать потребности клиентов и создавать более целенаправленные стратегии.

1.2.5 Современные тенденции (2010-е и после)

Современные CRM-системы стали широко использоваться в различных отраслях благодаря инновационным технологиям, таким как искусственный интеллект, аналитика данных, облачные решения и мобильные приложения. Это

позволяет бизнесу не только управлять данными о клиентах, но и предсказывать их потребности, что становится ключевым элементом конкурентоспособности.

1.2.6 Будущее CRM: интеллектуальная персонализация и интеграция

Прогнозируется, что будущее CRM будет сосредоточено на интеллектуальной персонализации, где системы будут использовать данные для предоставления клиентам еще более индивидуализированных взаимодействий. Интеграция CRM с другими технологическими решениями также будет нарастать, обеспечивая комплексный подход к управлению клиентским опытом.

Эволюция CRM свидетельствует о постоянном развитии и адаптации к изменяющимся условиям бизнеса, выделяя важность эффективного управления отношениями с клиентами для успеха компаний в современном мире.

1.3 Роль CRM в управлении клиентским опытом

1.3.1 Определение клиентского опыта и его значение

Клиентский опыт (Customer Experience, CX) представляет собой совокупность впечатлений и взаимодействий клиента с брендом или продуктом на протяжении всего жизненного цикла. Роль CRM в управлении клиентским опытом заключается в создании положительных и значимых взаимодействий, формирующих долгосрочные отношения.

1.3.2 Связь между CRM и CX

CRM и CX тесно взаимосвязаны. CRM предоставляет инструменты для сбора, анализа и использования данных, необходимых для понимания потребностей и предпочтений клиентов. Затем эти данные используются для улучшения каждого этапа взаимодействия с клиентами, формируя положительный клиентский опыт.

1.3.3 Персонализированные Взаимодействия

Роль CRM в управлении клиентским опытом проявляется в создании персонализированных стратегий взаимодействия. Используя данные о клиентах, CRM-системы позволяют предоставлять клиентам индивидуально адаптированные предложения, согласованные с их интересами и предпочтениями.

1.3.4 Анализ и Прогнозирование Поведения Клиентов

CRM не только реагирует на текущие потребности клиентов, но и предоставляет возможность анализа и прогнозирования их поведения. Это позволяет компаниям антиципировать потребности клиентов и предоставлять соответствующие услуги или продукты заранее.

1.3.5 Интеграция Каналов Общения

CRM играет ключевую роль в интеграции различных каналов общения с клиентами. Это позволяет клиентам переходить между онлайн и офлайн взаимодействиями, сохраняя при этом единое и согласованное восприятие бренда.

1.3.6 Отзывы и Обратная Связь Клиентов

CRM также является инструментом для сбора обратной связи от клиентов. Анализируя отзывы и мнения, компании могут вносить коррективы в свои стратегии, стремясь к непрерывному улучшению клиентского опыта.

1.3.7 Создание Долгосрочных Отношений

В целом, роль CRM в управлении клиентским опытом заключается в создании условий для долгосрочных и взаимовыгодных отношений между клиентами и брендом. Путем предоставления персонализированных, целенаправленных взаимодействий CRM способствует укреплению лояльности и повышению удовлетворенности клиентов.

1.4 Применение CRM в образовательной сфере

1.4.1 Особенности Образовательной Сферы и Необходимость CRM

Образовательная сфера характеризуется уникальными особенностями, такими как множество взаимодействий с учащимися, сложная структура учебного процесса и необходимость индивидуализированного подхода. Внедрение Customer Relationship Management (CRM) в учебные учреждения становится неотъемлемой частью для оптимизации взаимодействия с учениками и улучшения качества образования.

1.4.2 Анализ особенностей применения CRM в образовательных учреждениях

- Учет индивидуальных потребностей:

CRM в образовании позволяет учителям и администрации отслеживать индивидуальные потребности учеников, исходя из данных об успеваемости, интересах и стиле обучения.

- Эффективное управление классами и расписанием:

CRM помогает оптимизировать расписание, управлять группами и назначать занятия, учитывая предпочтения и особенности каждого ученика.

- Взаимодействие с родителями:

Встроенные в CRM средства коммуникации позволяют учителям и администрации эффективно взаимодействовать с родителями, предоставлять им информацию о успехах и прогрессе их детей.

- Учет динамики прогресса:

Системы CRM позволяют отслеживать динамику успеха и прогресса учеников, что облегчает принятие обоснованных решений для их развития.

1.4.3 Преимущества и вызовы внедрения CRM в образовательный процесс

- Улучшение учебного процесса:

Внедрение CRM в образовательные учреждения содействует более эффективному управлению образовательным процессом, поддерживая преподавателей и студентов в достижении лучших результатов.

- Персонализация обучения:

CRM позволяет создавать персонализированные образовательные программы, учитывая потребности и темпы усвоения информации у каждого ученика.

- Взаимодействие и связь:

Одним из ключевых преимуществ является улучшенное взаимодействие между учениками, преподавателями и администрацией. Это способствует открытости общения и быстрому реагированию на потребности студентов.

- Вызовы внедрения:

Внедрение CRM в образовательную среду может столкнуться с вызовами, такими как необходимость обучения персонала, интеграция существующих систем и обеспечение безопасности данных студентов.

1.4.4 Опыт успешного использования CRM в учебных учреждениях

- Улучшение успеваемости и мониторинг прогресса:

Одно из учебных заведений успешно использовало CRM для отслеживания успеваемости и предоставления персонализированной поддержки студентам, что привело к значительному улучшению результатов.

- Эффективное взаимодействие с родителями:

Другая организация сфокусировалась на использовании CRM для улучшения связи с родителями, что привело к увеличению вовлеченности и поддержки семейного образования.

Глава 2 Технологии и инструменты CRM

2.1 Обзор основных технологий и инструментов в CRM-системах

2.1.1 Базы данных и хранение информации:

Реляционные базы данных (например, MySQL, PostgreSQL) обеспечивают структурированное хранение данных о клиентах и их взаимодействии с системой CRM.

Облачные хранилища (например, Amazon S3, Google Cloud Storage) позволяют эффективно хранить и обрабатывать объемы данных.

2.1.2 Аналитика и обработка данных:

Инструменты аналитики данных (например, Tableau, Power BI) помогают проводить глубокий анализ информации, выявлять тенденции и формировать стратегии взаимодействия.

Big Data технологии (Hadoop, Apache Spark) применяются для обработки и анализа больших объемов данных в реальном времени.

2.1.3 Искусственный интеллект и машинное обучение:

Искусственный интеллект (AI) и машинное обучение (ML) используются для предсказания поведения клиентов, персонализации взаимодействия и автоматизации процессов (например, TensorFlow, scikit-learn).

2.1.4 Инструменты автоматизации процессов:

Платформы для автоматизации маркетинга и продаж (например, HubSpot, Salesforce Marketing Cloud) облегчают процессы взаимодействия с клиентами, отслеживают лиды и автоматизируют рутинные задачи.

2.2. Сравнительный анализ различных CRM-платформ

| | amoCRM | Битрикс24 | Мегаплан | Простой бизнес |
|-----------------------------|--|---|--|---|
| Интерфейс | Простой, стильный, интуитивно понятный интерфейс | Сложный и не наглядный интерфейс | Сложный интерфейс | Некрасивый дизайн интерфейса, но есть встроенный конструктор сайта |
| Функционал | Узкоспециализированная, ориентирована на продажи (функциональна) | Функционала много. Долго и трудно разбираться, неудобный таск-менеджер | Функционален, сложно разобраться | Функционален, но зависимость от приложения на Windows |
| Телефония | Интеграция с любой телефонией | Система работает со многими крупными компаниями. Есть возможность создавать виртуальные номера и АТС. | Интеграция с любой телефонией | Есть внутренняя бесплатная IP-телефония, есть возможность интегрировать свою или создать новую |
| Работа с документами | С помощью виджетов | Внутри сервиса, выглядит как соц сеть | Большое внимание к документации, есть свои готовые шаблоны | Автоматическая подготовка бухгалтерских и других документов, загрузка электронной подписи, печатей. Отличная работа с документами |

| | | | | |
|--|---|---|--|--|
| Аналитика | Наглядная аналитика, можно любую цифру вывести на рабочий стол | Руководитель видит над какими задачами в данный момент работают сотрудники и когда они их решают. Можно собрать статистику по задачам в конце месяца и оформить всё это в виде удобной инфографики. | Нельзя строить сложные отчеты: учет склада, перекрестный учёт. | Наглядная аналитика, модуль склада с аналитикой, движение финансов |
| Гибкость, возможность доработки | Интеграция с большим количеством сервисов, отличная возможность доработки | Интеграция с 1c, соц сетями, другими известными CRM и другими сервисами | Интеграция с другими сервисами (выбор не велик) | Не интегрируется с 1c, сложная интеграция в целом |
| Коммуникации между сотрудниками | Комментарии, чат, управление доступом, есть свой очень удобный мессенджер | Система выглядит, как соц сеть, сотрудники могут выкладывать посты и фотки, ставить лайки | Неудобное общение между сотрудниками | Широкая коммуникация, можно проводить видеоконференции |
| Облако | ✓ | ✓ | ✓ | ✓ |
| Коробка | ✗ | ✓ | ✓ | ✓ |
| Мобильная версия | ✓ | ✓ | ✓ | ✓ |

2.3 Особенности CRM для репетиторов

2.3.1 Учет расписания и занятий:

CRM для репетиторов должна предоставлять инструменты для эффективного учета расписания занятий и гибкой настройки графика работы.

2.3.2 Управление учениками и их прогрессом:

Функционал CRM должен включать возможности учета информации о каждом ученике, их успехах, слабых местах и динамике прогресса.

2.3.3 Ведение финансовых данных:

Интегрированные инструменты для учета финансовых операций, выставления счетов и мониторинга оплат, что является важным аспектом для репетиторов-предпринимателей.

2.3.4 Инструменты маркетинга и продвижения:

CRM должна обеспечивать возможности проведения маркетинговых кампаний, управления рекламой и привлечения новых учеников.

2.3.5 Аналитика и отчетность:

Наличие инструментов аналитики для оценки эффективности занятий, конверсии из лидов в учеников, а также другие ключевые метрики.

2.3.6 Персонализация учета учеников:

Возможность добавления персональных данных о каждом ученике, включая их учебные потребности, предпочтения и особенности.

2.3.7 Интегрированные инструменты обратной связи:

Средства для взаимодействия с учениками и их родителями, а также интегрированные средства обратной связи после каждого занятия.

2.3.8 Эффективное управление расписанием:

Инструменты для удобного планирования и редактирования расписания, а также уведомления и напоминания для учеников и репетиторов.

Глава 3 Разработка CRM системы для репетитора «ТИЧА»

3.1 Технологический стек и разработка CRM системы

3.1.1 VDS Сервер:

VDS сервер выделяет виртуальные вычислительные ресурсы специально для CRM. Это важно для обеспечения высокой производительности, масштабируемости и управления ресурсами в зависимости от потребностей приложения.

Mgnhost является хостинговым провайдером, предоставляющим услуги по размещению приложения в облаке. Это обеспечивает хранение данных CRM, их резервное копирование и обеспечивает высокую доступность веб-приложения.

| Id | Доменное имя | IP-адрес | Шаблон ОС |
|--|-----------------------|------------|--------------------|
| <input checked="" type="checkbox"/> 124978 | vds124978.mgnhost.com | 5.44.45.70 | Ubuntu-20.04-amd64 |

| Тариф | Дата-центр | Действует до | Состояние | Стоимость |
|----------|---------------------|--------------|-----------|--------------------|
| KVM-SSD2 | Дронтен, Нидерланды | 2024-03-19 | ⌚ Активен | 450.00 RUB / Месяц |

Рисунок 1 VDS сервер

Доступ к серверу осуществляется по SSH с ключом RSA (лучшая практика по подключению к удаленному серверу).

3.1.2 СУБД MySQL

MySQL служит в качестве реляционной базы данных для хранения и управления данными в CRM. Это важный компонент, обеспечивающий эффективное и структурированное хранение информации о репетиторах, учениках, расписании и финансах. В рамках данного проекта используется версия:

```
root@vds124978:~# mysql --version
mysql Ver 8.0.36-0ubuntu0.22.04.1 for Linux on x86_64 ((Ubuntu))
root@vds124978:~#
```

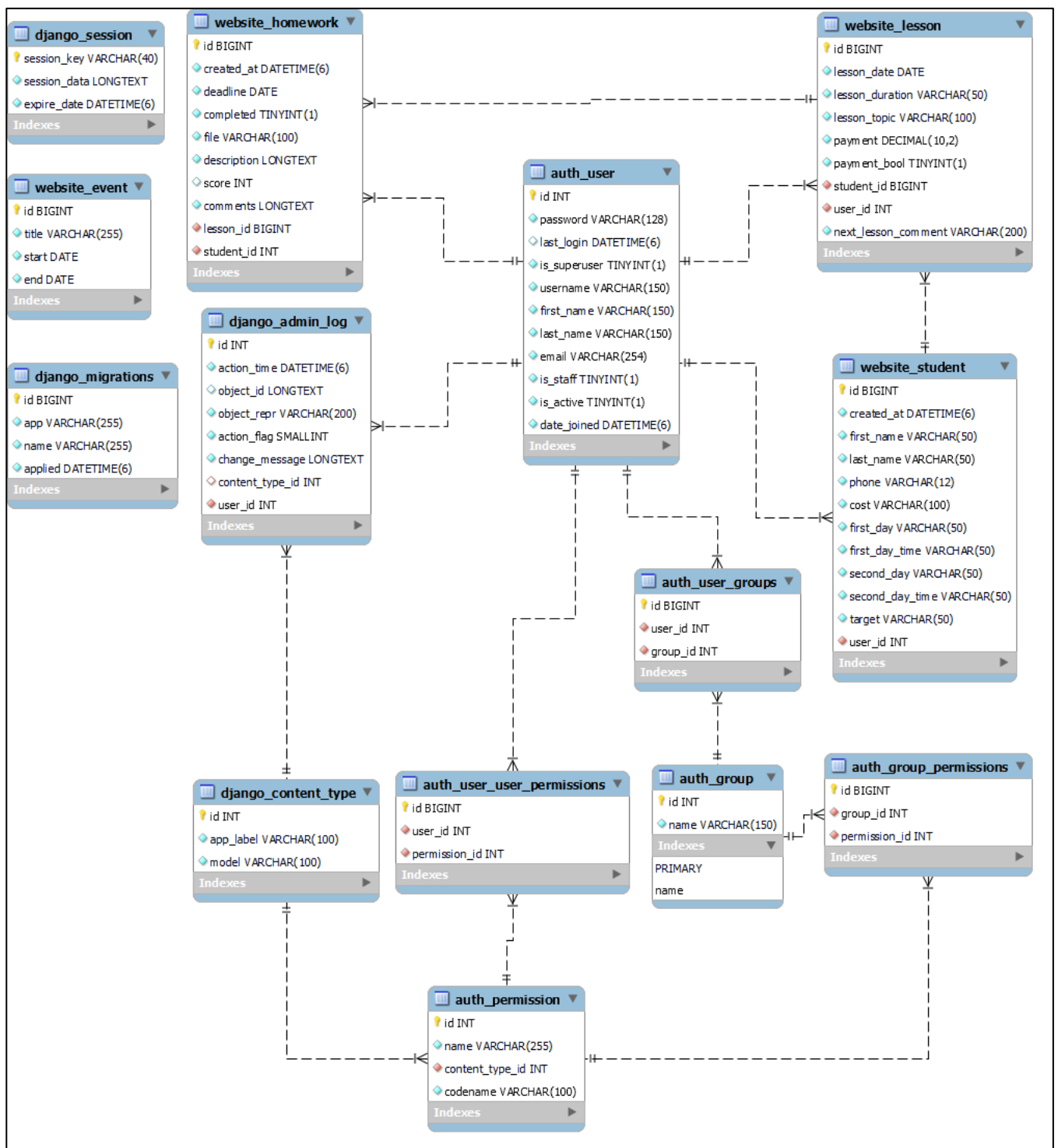


Рисунок 2 EER-диаграмма связей данных проекта

Краткое описание таблиц в БД:

1. auth_group:

- Поля: id, user_id, group_id
- Назначение: хранение информации о группах пользователей.

2. auth_group_permissions:

- Поля: id, group_id, permission_id
- Назначение: связывание групп пользователей с разрешениями (полномочиями) в системе.

3. auth_permission:

- Поля: id, name, content_type_id, codename
- Назначение: хранение информации о разрешениях, которые могут быть назначены пользователям или группам.

4. auth_user:

- Поля: id, password, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined
- Назначение: содержит основную информацию о пользователях системы.

5. auth_user_groups:

- Поля: id, user_id, group_id
- Назначение: связывание пользователей с группами.

6. auth_user_user_permissions:

- Поля: id, user_id, permission_id
- Назначение: связывание пользователей с индивидуальными разрешениями.

7. django_admin_log:

- Поля: id, action_time, object_id, object_repr, action_flag, change_message, content_type_id, user_id
- Назначение: журналирование действий администраторов в системе Django.

8. django_content_type:

- Поля: id, app_label, model
- Назначение: хранение информации о типах содержимого (моделях) в системе Django.

9. django_migrations:

- Поля: id, app, name, applied
- Назначение: хранение информации о миграциях базы данных Django.

10. django_session:

- Поля: session_key, session_data, expire_date

- Назначение: хранение данных сессий для веб-приложений Django.
- 11. website_event:**
- Поля: id, title, start, end
 - Назначение: содержит информацию о событиях (уроках, встречах, переносах и т.д.).
- 12. website_homework:**
- Поля: id, created_at, deadline, completed, file, description, score, comments, lesson_id, student_id
 - Назначение: содержит информацию о домашних заданиях, связанных с конкретными уроками (на данный момент не используется).
- 13. website_lesson:**
- Поля: id, lesson_date, lesson_duration, lesson_topic, payment, payment_bool, student_id, user_id, next_lesson_comment
 - Назначение: хранение информации о уроках.
- 14. website_student:**
- Поля: id, created_at, first_name, last_name, phone, cost, first_day, first_day_time, second_day, second_day_time, target, user_id
 - Назначение: содержит информацию о студентах, связанных с пользователями системы.

Создание таблиц в базе данных Django осуществляется с использованием миграций. Миграции — это способ изменения структуры базы данных в соответствии с изменениями в моделях Django. Когда определяются модели (классы) в приложении Django, затем создаются и применяются миграции, чтобы изменения были отражены в базе данных.

Процесс создания этих таблиц может выглядеть следующим образом: в Django-приложении определены модели для каждой таблицы. Пример определения модели для таблицы `auth_user` может выглядеть так:

```
from django.db import models
from django.contrib.auth.models import AbstractUser
class CustomUser(AbstractUser):
    # Дополнительные поля, если нужны
    pass
```

Можно по-разному определить модели для каждой таблицы. После определения моделей выполните команду создания миграции:

```
python manage.py makemigrations
```

Это создаст файл миграции в приложении. Затем выполните команду применения миграции:

```
python manage.py migrate
```

Это применит изменения к базе данных, создав необходимые таблицы.

Процесс повторяется для каждой модели в приложении. Django автоматически создает таблицы, связи и индексы на основе моделей и их полей.

Обратите внимание, что модели могут также использовать атрибуты, такие как `ForeignKey` и `OneToOneField`, чтобы определить внешние ключи и связи между таблицами. Когда эти миграции применяются, Django автоматически создает соответствующие внешние ключи в базе данных.

Резервное копирование осуществляется 1 раз в неделю и осуществляется с использованием MySQL Workbench:

- 1) Откройте MySQL Workbench и подключитесь к серверу.
- 2) Выберите вашу базу данных в окне "Navigator".
- 3) Перейдите во вкладку "Server" в верхнем меню.
- 4) Выберите "Data Export".

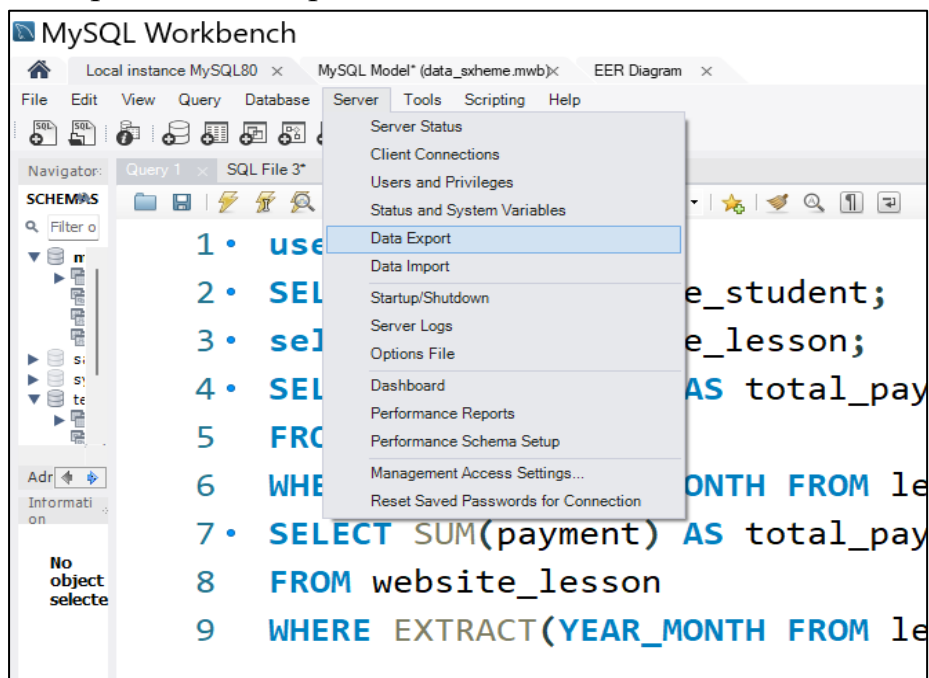


Рисунок 3 Меню Server

- 5) Выберите базу данных, которую вы хотите экспортировать.
- 6) Настройте параметры экспорта по желанию (например, выберите место для сохранения файла).
- 7) Нажмите "Start Export".

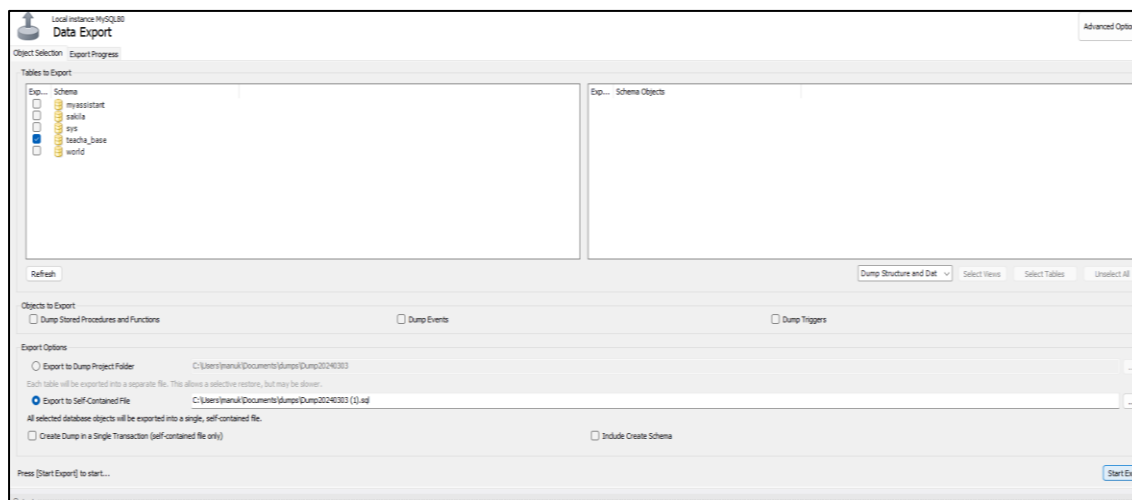


Рисунок 4 Меню Data Export

MySQL Workbench создаст резервную копию базы данных в выбранном месте.

3.1.3 Nginx (Веб-Сервер)

Nginx — это высокопроизводительный веб-сервер и прокси-сервер, который также может выполнять функции обратного прокси, обработки нагрузки и балансировки нагрузки. Он обладает эффективной архитектурой, разработанной с учетом масштабируемости и обработки большого количества соединений.

Вот несколько ключевых характеристик и возможностей Nginx:

1. Эффективность и Производительность:

Nginx разработан с упором на эффективное использование ресурсов, что позволяет ему обслуживать множество одновременных соединений с минимальным потреблением ресурсов.

2. Обработка Статики и Динамики:

Эффективно обрабатывает статический и динамический контент. Может быть использован для обслуживания статических файлов напрямую, а также в качестве прокси для приложений, обрабатывающих динамический контент.

3. Проксирование и Балансировка Нагрузки:

Nginx может работать в качестве прокси-сервера, перенаправляя запросы на другие серверы. Он также поддерживает балансировку нагрузки между несколькими серверами для обеспечения равномерного распределения нагрузки.

Настройка Nginx происходила следующим образом:

Создание файла конфигурации для сайта:

```
root@vds124978:~# vim /etc/nginx/sites-available/teacha_crm
```

Создание базовой конфигурации

```
root@vds124978:~# vim /etc/nginx/sites-available/teacha_crm
server {
    listen 80;
    server_name 185.180.109.118;
    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/Teacha/TEACHA_CRM/teacha_crm;
    }
    location /media/ {
        root <directory_root>;
    }
    location /admin/static/ {
        root /home/Teacha/TEACHA_CRM;
    }
    location / {
        include proxy_params;
```

```
    proxy_pass http://unix:/home/Teacha/TEACHA_CRM/server.sock;
}
}
```

Создание символической ссылки на конфигурацию:

```
root@vds124978:~# ln -s /etc/nginx/sites-available/teacha_crm
/etc/nginx/sites-enabled/teacha_crm
```

Проверка файла конфигурации:

```
root@vds124978:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Перезагрузка службы Nginx

```
root@vds124978:~# sudo service nginx restart
```

3.1.4 Gunicorn (WSGI HTTP-сервер):

Gunicorn является HTTP-сервером, специально созданным для обслуживания веб-приложений Django. Он обеспечивает высокую производительность, поддерживает множество соединений и эффективно работает с веб-приложениями Python.

Файл конфигурации Gunicorn:

```
root@vds124978:~#vim
/etc/systemd/system/teacha_crm.gunicorn.service
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=root
```



```
Group=www-data
WorkingDirectory=/home/Teacha/TEACHA_CRM/teacha_crm
ExecStart=/home/Teacha/TEACHA_CRM/myenv/bin/gunicorn      --access-
logfile          -          --workers          3          --bind
unix:/home/Teacha/TEACHA_CRM/server.sock
teacha_crm.wsgi:application

[Install]
WantedBy=multi-user.target
```

Запуск службы gunicorn:

```
root@vds124978:~# systemctl start teacha_crm.gunicorn
```

Добавляем автозапуск службы:

```
root@vds124978:~# systemctl enable teacha_crm.gunicorn
```

Проверяем работу:

```
root@vds124978:~# systemctl status teacha_crm.gunicorn
● teacha_crm.gunicorn.service - gunicorn daemon
   Loaded: loaded
           (/etc/systemd/system/teacha_crm.gunicorn.service; enabled; vendor
           preset: enabled)
   Active: active (running) since Mon 2024-02-19 07:41:23 MSK; 1
           week 6 days ago
     Main PID: 679 (gunicorn)
        Tasks: 4 (limit: 1063)
       Memory: 98.2M
          CPU: 14min 53.382s
    CGroup: /system.slice/teacha_crm.gunicorn.service
            └─679 /home/Teacha/TEACHA_CRM/myenv/bin/python3
               /home/Teacha/TEACHA_CRM/myenv/bin/gunicorn --access-logfile - --
               workers 3 --bind un>
```

3.1.5 Linux (Операционная Система):

Linux, как серверная операционная система, предоставляет стабильность, безопасность и высокую производительность для веб-приложения. Многие современные веб-серверы и фреймворки предпочитают использовать Linux в производственных средах.

3.1.6 Python (язык программирования):

Python, как основной язык программирования, обеспечивает легкость разработки, читаемость кода и широкий спектр библиотек для решения различных задач. В контексте CRM, Python используется для бизнес-логики, взаимодействия с внешними сервисами и обработки данных.

Для организации бизнес-логики в рамках проекта использованы следующие библиотеки:

```
asgiref==3.7.2
Django==4.2.5
mysql==0.0.3
mysql-connector==2.2.9
mysql-connector-python==8.1.0
mysqlclient==2.2.0
protobuf==4.21.12
sqlparse==0.4.4
tzdata==2023.3
pytz~=2023.3.post1
python-dateutil~=2.8.2
```

Данные библиотеки зафиксированы в виртуальном окружении в файле requirements.txt, созданным при старте разработки проекта. Все дальнейшие импорты организованы из стандартных библиотек яп. Python и фреймворка Django.

Реализация бизнес-логики отражена по пути teacha_crm\website\views_my. В данной директории содержатся модули, поддерживающие регистрацию,

авторизацию, все CRUD (create, read, update, delete) операции по работе с базой данных. В директории существуют поддиректории зон ответственности каждого представления. Остановимся на некоторых.

Логика работы стартовых страниц:

Директория: `teacha_crm\website\views_my\main\main_view.py`

1. `home(request):`

Описание: это представление отображает домашнюю страницу пользователя после входа в систему. Отображает текущее время, информацию о студентах пользователя, занятиях в текущий день, и общую сумму платежей за текущий месяц. Поддерживает пагинацию для списка студентов.

Действия: фильтрует студентов текущего пользователя, определяет текущий день недели на русском, рассчитывает общую сумму платежей за текущий месяц.

Поддерживает пагинацию для списка студентов.

2. `login_user(request):`

Описание: обрабатывает запросы на вход в систему. Пользователь вводит имя пользователя и пароль. Если данные верны, пользователь входит в систему, иначе выводится сообщение об ошибке.

Действия: аутентификация пользователя, вход в систему и перенаправление на главную страницу, вывод сообщения об ошибке при неудачной попытке входа.

```
def login_user(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username,
                             password=password)
        if user is not None:
            login(request, user)
```

```
        messages.success(request, f"Успешная авторизация,  
{request.user.first_name}"  
                                f" {request.user.last_name}!")  
        return redirect('index')  
        messages.success(request, "Ошибка, попробуйте вновь")  
        return redirect('login')  
    return render(request, 'main/login.html')
```

3. logout_user(request):

Описание: разлогинивает текущего пользователя и перенаправляет на домашнюю страницу. Выводит сообщение об успешном выходе из системы.

Действия: проверка аутентификации пользователя, разлогинивание пользователя и перенаправление.

4. register(request):

Описание: обрабатывает запросы на регистрацию нового пользователя. Пользователь вводит данные, форма проверяется на валидность, и в случае успеха происходит вход в систему.

Действия: проверка метода запроса (POST), создание формы регистрации, проверка валидности формы и создание нового пользователя, вход в систему и перенаправление на главную страницу.

```
def register(request):  
    if request.method == 'POST':  
        form = SignUpForm(request.POST)  
        if form.is_valid():  
            form.save()  
            # Авторизация и вход  
            username = form.cleaned_data['username']  
            password = form.cleaned_data['password1']  
            user=authenticate(username=username,password=password)  
            login(request, user)
```

```
        messages.success(request, f"{username} - Вы успешно
зарегистрировались, добро пожаловать")
        return redirect('home')
    else:
        form = SignUpForm()
        return render(request, 'profile/register.html', {'form':
form})
    return render(request, 'profile/register.html', {'form':
form})
```

5. history(request):

Описание: отображает историю уроков пользователя, сгруппированную по месяцам. Показывает список уроков, проведенных за каждый месяц.

Действия: фильтрация всех уроков текущего пользователя, группировка уроков по месяцам.

6. showDashboard(request):

Описание: отображает дашборд с общей статистикой пользователя, такой как количество студентов, количество уроков, общая длительность уроков и другие данные.

Действия: фильтрация студентов текущего пользователя, расчёт статистики: общее количество студентов, уроков и общая длительность уроков.

Логика работы с уроками:

Директория: teacha_crm\website\views_my\lesson\lesson_view.py

Данный модуль содержит представления (views) для управления занятиями в приложении. Здесь реализованы функции для добавления, редактирования, удаления и просмотра занятий.

1. `add_lessons(request):`

Описание: добавляет информацию о пройденном уроке в систему. Отображает форму, где пользователь может внести данные о занятии, такие как дата, время, длительность, студент и оплата.

Действия: использует форму `show_lessons`, проверяет метод запроса (`post`), при успешном внесении данных перенаправляет пользователя на домашнюю страницу.

2. `lesson_record(request, lesson_id):`

Описание: позволяет пользователю редактировать информацию о пройденном уроке. Отображает форму с текущими данными урока, которые можно изменить.

Действия: получает урок с помощью `get_object_or_404`, проверяет метод запроса (`post`), при успешном редактировании данных урока перенаправляет пользователя на страницу редактирования урока.

3. `delete_lesson(request, lesson_id):`

Описание: удаляет запись о пройденном уроке. Предоставляет пользователю подтверждение удаления.

Действия: получает урок с помощью `get_object_or_404`, проверяет метод запроса (`post`), при подтверждении удаления урока удаляет запись и перенаправляет пользователя на домашнюю страницу.

Логика работы с записями об учениках

Директория: `teacha_crm\website\views_my\student\student_view.py`

Этот модуль содержит представления (`views`) для управления студентами в приложении. В нем реализованы функции для добавления, редактирования, удаления и просмотра информации о студентах.

1. `add_student(request):`

Описание: добавляет нового студента в систему. Отображает форму, где пользователь может внести данные о студенте, такие как имя, фамилия, контактная информация и дополнительные заметки.

Действия: использует форму `Add_record`, проверяет метод запроса (POST), при успешном внесении данных перенаправляет пользователя на домашнюю страницу.

2. `delete_student(request, pk):`

Описание: удаляет запись о студенте, предоставляет пользователю подтверждение удаления.

Действия: получает студента с помощью `Student.objects.get(id=pk)`, проверяет метод запроса (POST), при подтверждении удаления студента удаляет запись и перенаправляет пользователя на домашнюю страницу.

3. `update_student(request, pk):`

Описание: позволяет пользователю редактировать информацию о студенте. Отображает форму с текущими данными студента, которые можно изменить.

Действия: получает студента с помощью `Student.objects.get(id=pk)`, проверяет метод запроса (POST), при успешном редактировании данных студента перенаправляет пользователя на домашнюю страницу.

4. `record_student(request, pk):`

Описание: отображает записи об уроках для выбранного студента. Если студент принадлежит текущему пользователю, выводит информацию о студенте и список его уроков.

Действия: получает студента с помощью `Student.objects.filter(user=current_user, id=pk).first()`, если студент найден, получает его уроки и отображает их, в случае ошибки выводит сообщение и перенаправляет на домашнюю страницу.

Логика работы со статистикой

Директория: teacha_crm\website\views_my\statistic\get_stat.py

Данный модуль предназначен для получения статистических данных о занятиях студентов и их оплатах. Он рассчитывает общую сумму оплаты за месяц, сравнивает эту сумму с общей стоимостью всех занятий и выводит информацию о прогрессе в достижении цели.

1. get_stat(request)

Описание: функция получает статистику по оплатам занятий за текущий месяц для каждого студента.

Действия: использует текущего пользователя из запроса, получает оплаты занятий для каждого студента, составляет списки с именами студентов и соответствующими им оплатами, рассчитывает сумму оплаты для каждого дня недели, определяет количество дней в месяце для каждого дня недели, рассчитывает сумму оплаты за месяц и сравнивает её с общей стоимостью занятий.

```
""" Описание модуля """
from calendar import monthrange
from datetime import date, timedelta
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.db.models import Sum
from ...models import Student, Lesson

day_translation = {
    'Monday': 'Понедельник',
    'Tuesday': 'Вторник',
    'Wednesday': 'Среда',
    'Thursday': 'Четверг',
    'Friday': 'Пятница',
```



```

    'Saturday': 'Суббота',
    'Sunday': 'Воскресенье',
}

@login_required
def get_stat(request):
    """ Описание функции """
    user = request.user

    student_payments = Lesson.objects.filter(user=user).values('student__first_name', 'student__last_name').annotate(
        total_payment=Sum('payment'))
    student_names = [f"{entry['student__first_name']} " \
                     f"{entry['student__last_name']}" for entry in student_payments]
    total_payments = [int(entry['total_payment']) for entry in student_payments]

    # ----Пайчарт для сравнения заработано и сколько осталось
    year = date.today().year
    month = date.today().month
    user = request.user
    students = Student.objects.filter(user=user)
    day_payments = {
        'Понедельник': 0,
        'Вторник': 0,
        'Среда': 0,
        'Четверг': 0,
        'Пятница': 0,
        'Суббота': 0,
        'Воскресенье': 0,
    }

    day_of_week_to_day_in_month = {

```

```

    'Понедельник': 1,
    'Вторник': 2,
    'Среда': 3,
    'Четверг': 4,
    'Пятница': 5,
    'Суббота': 6,
    'Воскресенье': 7,
}

days_count = {day: 0 for day in range(7)} # 0 - Понедельник,
1 - Вторник, и т.д.

# Переберите записи и увеличьте суммы payment для
соответствующих дней недели
for student in students:
    first_day_of_week = student.first_day # Название дня
недели (например, 'Вторник')
    second_day_of_week = student.second_day # Название дня
недели (например, 'Вторник')
    # Получите соответствующее число дня в месяце
    first_day_in_month =
day_of_week_to_day_in_month.get(first_day_of_week)
    second_day_in_month =
day_of_week_to_day_in_month.get(second_day_of_week)

    if first_day_in_month is not None:
        # Увеличьте сумму payment для этого дня недели
        day_payments[first_day_of_week] += int(student.cost)
        # print(day_payments)
    if second_day_in_month is not None:
        # Увеличьте сумму payment для этого дня недели
        day_payments[second_day_of_week] += int(student.cost)
        # print(day_payments)

# Перевод словаря с Понедельник Вторник в 0 1 в тд

```

```

new_key = [0, 1, 2, 3, 4, 5, 6]
day_payments = dict(zip(new_key, list(day_payments.values()))))
# -----
-----

# Находим первый и последний дни месяца
first_day = date(year, month, 1)
last_day = date(year, month, monthrange(year, month)[1])

# Перебираем все дни в месяце и увеличиваем счетчик
соответствующего дня недели
current_day = first_day
while current_day <= last_day:
    day_of_week = current_day.weekday() # 0 - Понедельник, 1
- Вторник, и т.д.
    days_count[day_of_week] += 1
    current_day += timedelta(days=1)
    # Нахождение полной зарплаты за месяц
cost_per_month = {day: days_count[day] * day_payments[day] for
day in days_count}

# Получить сумму всех занятий за текущий месяц
total_payment_true = Lesson.objects.filter(user=user,
                                             lesson_date__gte=f
first_day,
                                             lesson_date__lt=la
st_day,
                                             payment_bool=True
                                             ).aggregate(total=
Sum('payment'))['total'] or 0.00
total_payment_false = Lesson.objects.filter(user=user,
                                             lesson_date__gte=
first_day,
                                             lesson_date__lt=l
ast_day,

```

```

payment_bool=False
).aggregate(total
=Sum('payment'))['total'] or 0.00

context = {
    'student_names': student_names,
    'total_payments': total_payments,
    'goal': (sum(cost_per_month.values()) -
int(total_payment_true) - int(total_payment_false)),
    'current': int(total_payment_true),
    'not_paid': int(total_payment_false),
}
return render(request, 'profile/statistic.html', context)

```

3.1.7 Django (Python Web Framework)

Django является высокоуровневым веб-фреймворком, написанным на языке Python. Он обеспечивает разработку масштабируемых веб-приложений, включая удобное взаимодействие с базой данных, обработку HTTP-запросов и множество готовых инструментов для ускорения разработки.

Создание проекта:

Установка Django:

Установите Django, выполнив команду в терминале.

```
pip install Django
```

Создание проекта:

Используйте команду

```
django-admin startproject projectname
```

для создания нового проекта Django.

Структура проекта:

manage.py: командный файл для управления различными аспектами проекта (например, запуск сервера, создание миграций).

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
        'teacha_crm.settings')
    try:
        from django.core.management import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed
and "
            "available on your PYTHONPATH environment variable?
Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

teacha_crm/: основная директория проекта.

settings.py: настройки проекта, такие как база данных, приложения, маршрутизация URL и другие. Некоторые настройки, связанные с проектом:

```
ALLOWED_HOSTS = ['*']
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'website'
]
ROOT_URLCONF = 'teacha_crm.urls'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'teacha_base',
        'USER': 'root',
        'PASSWORD': password,
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]
LOGIN_URL = 'login' # URL-адрес страницы входа
LOGIN_REDIRECT_URL = 'index' # URL-адрес после успешной аутентификации
LOGOUT_REDIRECT_URL = 'login' # URL-адрес после выхода
```

urls.py: Определение URL-маршрутов для обработки запросов.

```
urlpatterns = [
    path('home/', main_view.home, name='home'),
    path('', main_view.login_user, name='login'),
    path('logout/', main_view.logout_user, name='logout'),
    path('register/', main_view.register, name='register'),
    path('record/<int:pk>', student_view.record_student,
name='record'),
    path('delete_record/<int:pk>', student_view.delete_student,
name='delete_record'),
    path('add_record/', student_view.add_student, name='add_record'),
    path('update_record/<int:pk>', student_view.update_student,
name='update_record'),
    path('calendar/', calendar_view.calendar, name='calendar'),
    path('add_event', calendar_view.all_events, name='add_event'),
    path('update', calendar_view.update, name='update'),
    path('remove', calendar_view.remove, name='remove'),
    path('all_events', calendar_view.all_events,
name='all_events'),
    path('profile/', profile_view.profile_view, name='profile'),
    path('edit_profile/', profile_view.edit_profile_view,
name='edit_profile'),
    path('delete_profile/', profile_view.delete_profile,
name='delete_profile'),
    path('statistic/', get_stat, name='statistic'),
    path('add_lessons/', lesson_view.add_lessons,
name='add_lessons'),
    path('record_lesson/<int:lesson_id>',
lesson_view.lesson_record, name='lesson_record'),
    path('delete_lesson/<int:lesson_id>',
lesson_view.delete_lesson, name='delete_lesson'),
    path('password_reset/', auth_views.PasswordResetView.as_view(
), name='password_reset'),
```

```
path('password_reset/done/',
auth_views.PasswordResetDoneView.as_view(),name='password_reset_d
one'),
    path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfi
rmView.as_view(), name='password_reset_confirm'),
    path('reset/done/',auth_views.PasswordResetCompleteView.as_v
iew(), name='password_reset_complete'),
    path('history/', main_view.history, name='history'),
    path('dashboard/', main_view.showDashboard, name='index'),
]
```

wsgi.py: Конфигурация WSGI для развертывания проекта на сервере.

Создание приложений:

Используйте команду для создания нового приложения Django.

```
python manage.py startapp appname
```

Каждое приложение имеет свою собственную структуру, включая файлы для представлений, шаблонов, статических файлов и моделей.

Работа с фреймворком:

Модели:

Определение структуры базы данных с использованием моделей Django в файле models.py каждого приложения.

Файл models.py

```
from django.db import models
from django.contrib.auth.models import User
# База данных по студентам
class Student(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
default=None)
    created_at = models.DateTimeField(auto_now=True)
```



```

    first_name = models.CharField(max_length=50, default="default
title")
    last_name = models.CharField(max_length=50, default="default
title")
    phone = models.CharField(max_length=12, default="default
title")
    cost = models.CharField(max_length=100, default="default
title")
    target = models.CharField(max_length=50, default="default
title")
    first_day = models.CharField(max_length=50, default="default
title")
    first_day_time = models.CharField(max_length=50,
default="default title")
    second_day = models.CharField(max_length=50, default="default
title")
    second_day_time = models.CharField(max_length=50,
default="default title")

objects = models.Manager()

def __str__(self):
    return f"{self.first_name} {self.last_name}"

# База данных по урокам студентов
class Lesson(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
default=None)
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    lesson_date = models.DateField()
    lesson_duration = models.CharField(max_length=50,
default="default title")
    lesson_topic = models.CharField(max_length=100,
default="default title")

```

```

    payment = models.DecimalField(max_digits=10, decimal_places=2,
default=0.00)
    payment_bool = models.BooleanField(default=False)
    next_lesson_comment = models.CharField(max_length=200,
default=None)

objects = models.Manager()

def __str__(self):
    return f"Lesson for {self.student.first_name}
{self.student.last_name} on {self.lesson_date}"

# Модель для календаря
class Event(models.Model):
    title = models.CharField(max_length=255)
    start = models.DateField()
    end = models.DateField()
    objects = models.Manager()

    def __str__(self):
        return self.title

```

Выполнение миграций для создания или обновления схемы базы данных.

Создание представлений в файле `views.py`, определяющих, как обрабатывать запросы и возвращать ответы. Все представления разбиты по группам по логике влияния на содержимое базы данных.

Шаблоны:

Создание шаблонов (HTML-файлов) для отображения данных и формирования интерфейса в папке `templates` каждого приложения.

В рамках CRM системы организована группировка шаблонов по логике применения и хранятся в директории `teacha_crm\website\templates`. Данная директория включает в себя поддиректории:

```
teacha_crm\website\templates\lesson
teacha_crm\website\templates\main
teacha_crm\website\templates\profile
teacha_crm\website\templates\student
```

Остановимся подробнее на организации шаблонов.

Шаблоны, содержащиеся в директории `teacha_crm\website\templates\main`

```
teacha_crm\website\templates\main\base.html
teacha_crm\website\templates\main\dashboard.html
teacha_crm\website\templates\main\history.html
teacha_crm\website\templates\main\home.html
teacha_crm\website\templates\main\login.html
teacha_crm\website\templates\main\navbar.html
```

1) `teacha_crm\website\templates\main\base.html`

Этот код представляет собой базовый HTML-шаблон для веб-страницы, который используется в Django-проекте. Давайте рассмотрим его основные элементы:

`{% load static %}`: этот тег загружает статические файлы, такие как CSS и JavaScript, используя механизм статических файлов Django.

Подключение стилей и скриптов:

`{% static 'css/styles.css' %}`: загружает стилевой файл 'styles.css' из каталога 'css' в папке 'static'.

`{% static 'lib/noty.css' %}`: загружает стилевой файл 'noty.css' из каталога 'lib' в папке 'static'.

`<script src="{% static 'lib/noty.js' %}"></script>`: загружает JavaScript-библиотеку 'noty.js'.

`<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>` и `<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/4.4.0/chart.min.js"></script>`: загружают библиотеки для работы с графиками.

`<meta charset="utf-8">`: устанавливает кодировку документа.

`<meta name="viewport" content="width=device-width, initial-scale=1">`: определяет параметры отображения контента на мобильных устройствах.

`<title> TEACHA_CRM</title>`: устанавливает заголовок страницы.

Подключение Bootstrap:

`<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384 T3c6CoI6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">`: загружает файл стилей Bootstrap.

`{% include 'main/navbar.html' %}`: вставляет содержимое файла 'navbar.html', который, вероятно, содержит навигационное меню.

`<div class="container">`: определяет контейнер, в котором будет размещено основное содержимое страницы.

Обработка сообщений:

`{% if messages %}`: проверяет наличие сообщений для вывода.

`{% for message in messages %}`: перебирает сообщения для отображения в предупреждениях.

`{% block content %} {% endblock %}`: определяет блок контента, который может быть расширен в дочерних шаблонах.

Подключение Bootstrap JavaScript:

```
<script src=https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js
integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDf
L" crossorigin="anonymous"></script>
```

: загружает файл JavaScript Bootstrap.

Заключительные сведения и футер.

Этот HTML-шаблон создан для использования в Django-проекте и включает основные элементы для построения веб-страницы.

2) teacha_crm\website\templates\main\home.html

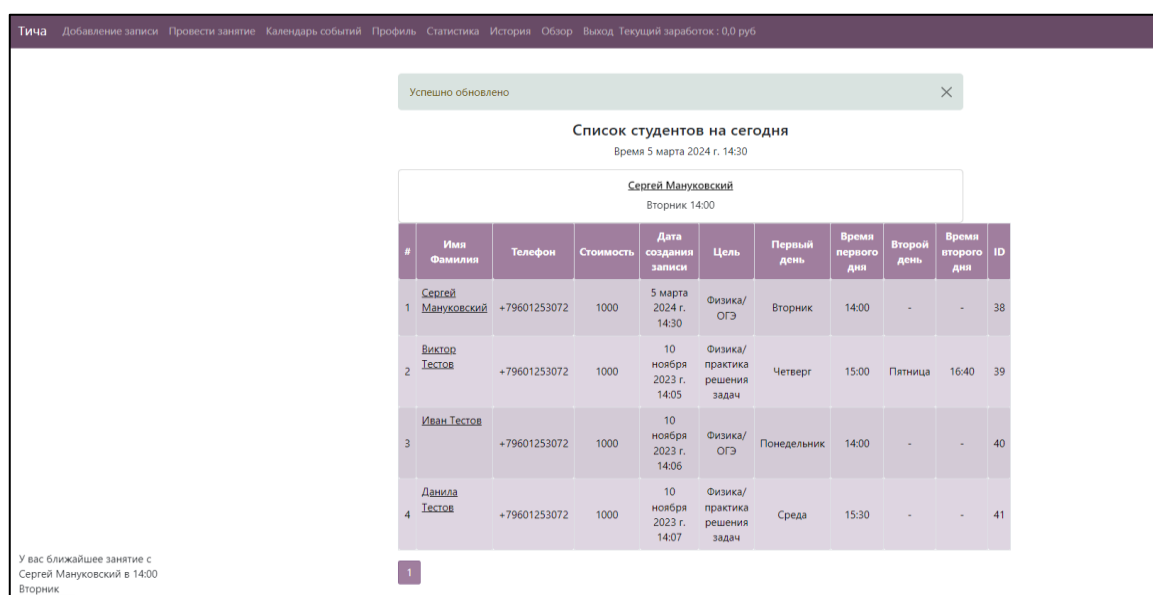


Рисунок 5 Шаблон home.html

Этот код представляет собой Django-шаблон (template), использующий расширение (extends) для базового шаблона 'main/base.html'. Давайте рассмотрим ключевые элементы этого шаблона:

{% extends 'main/base.html' %}: оповещает Django о том, что этот шаблон расширяет ('extends') базовый шаблон 'main/base.html'. Весь контент, определенный в базовом шаблоне, будет включен в этот шаблон.

`{% load static %}`: загружает статические файлы, такие как CSS и JavaScript, используя механизм статических файлов Django.

`{% block content %} ... {% endblock %}`: определяет блок контента, который может быть заполнен в дочерних шаблонах. Весь контент между этими тегами будет вставлен в соответствующее место блока 'content' в базовом шаблоне.

Подключение стилей:

`<link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}" />`:
Загружает стилевой файл 'styles.css' из каталога 'css' в папке 'static'.

Карусель студентов:

Использует Bootstrap Carousel для отображения списка студентов на текущей неделе.

Использует Django-шаблонные теги для вставки данных о студентах из переменной 'records_week'.

Таблица студентов:

Отображает таблицу со списком студентов.

Использует Django-шаблонные теги для вставки данных о студентах из переменной 'records'.

Имеет кнопки сортировки по различным столбцам, использующие функцию 'sortTable'.

Пагинация: используется для разбиения списка студентов на страницы, динамически формирует навигацию для переключения между страницами.

Передача данных в JavaScript:

В цикле передаются данные о ближайших занятиях студентов для отображения уведомлений.

Для этого используется создание HTML-элементов с атрибутами, содержащими необходимые данные.

Используется скрипт JavaScript для создания уведомлений с использованием библиотеки Noty.

Скрипты JavaScript:

Подключает скрипты для сортировки таблицы и создания уведомлений.

Календарь:

Использует библиотеку FullCalendar для отображения календаря событий.

Динамически обновляет события в календаре при их добавлении, изменении и удалении.

Toastr Success Code: используется библиотека Toastr для вывода всплывающих уведомлений об успешных событиях.

3) `teacha_crm\website\templates\main\login.html`

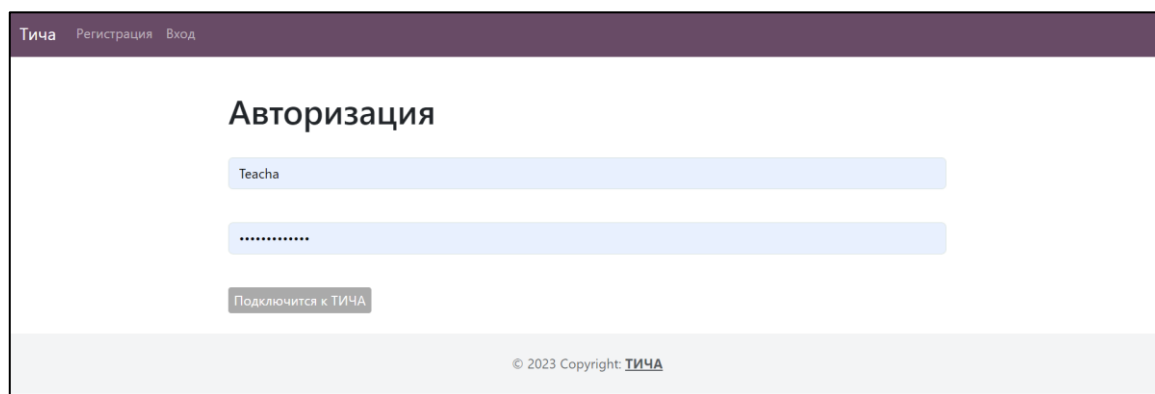
The image shows a web browser window displaying a login page. At the top, there is a dark purple header bar with the text 'Тича' followed by 'Регистрация' and 'Вход' in a lighter color. The main content area has a white background. In the center, the word 'Авторизация' is displayed in a bold, dark font. Below it, there are two light blue input fields. The first field contains the text 'Teacha'. The second field contains a series of dots, indicating a password. Below these fields is a button with a dark grey background and the text 'Подключится к ТИЧА' in white. At the bottom of the page, there is a light grey footer bar containing the text '© 2023 Copyright: ТИЧА'.

Рисунок 6 Шаблон login.html

Этот Django-шаблон предназначен для отображения формы авторизации. Давайте рассмотрим основные элементы шаблона:

`{% extends 'main/base.html' %}`: оповещает Django о том, что этот шаблон расширяет базовый шаблон 'main/base.html'. Это означает, что весь контент, определенный в базовом шаблоне, будет включен в этот.

`{% load static %}`: загружает статические файлы, такие как CSS и JavaScript, используя механизм статических файлов Django.

`{% block content %} ... {% endblock %}`: определяет блок контента, который может быть заполнен в дочерних шаблонах. Весь контент между этими тегами будет вставлен в соответствующее место блока 'content' в базовом шаблоне.

`<h1>Авторизация</h1>`: заголовок страницы, указывающий на то, что форма предназначена для авторизации.

Форма авторизации:

`method="POST"`: указывает, что данные формы будут отправлены методом POST.

`action="{% url 'login' %}"`: указывает URL, на который будет отправлен запрос при отправке формы.

`{% csrf_token %}`: вставляет токен CSRF для обеспечения безопасности передачи данных.

Поля формы:

`name="username"`: Поле для ввода имени пользователя.

`name="password"`: Поле для ввода пароля.

Кнопка отправки формы:

`type="submit"`: Определяет, что это кнопка отправки формы.

`class="btn btn-primary"`: Добавляет стили Bootstrap для кнопки.

Этот шаблон создан для страницы авторизации и предоставляет минимальные элементы, необходимые для ввода имени пользователя и пароля, а также кнопку для отправки данных на сервер. Весь этот контент будет вставлен в блок 'content' базового шаблона 'main/base.html'.

4) teacha_crm\website\templates\main\navbar.html

Рисунок 7 Шаблон navbar.html

Этот Django-шаблон представляет собой навигационную панель (navbar), которая используется для навигации по различным разделам веб-приложения. Давайте разберем основные элементы:

`{% load static %}`: загружает статические файлы, такие как CSS и JavaScript, используя механизм статических файлов Django.

Ссылки на стили:

Подключается файл стилей 'css/styles.css'.

Подключается файл стилей для иконок из библиотеки Font Awesome.

Навигационная панель (`<nav class="navbar navbar-expand-lg navbar-dark fixed-top">`):

`navbar-expand-lg`: Указывает, что навигационная панель будет расширяться для больших экранов.

`navbar-dark`: Указывает на использование темной темы для навигационной панели.

`fixed-top`: Закрепляет навигационную панель вверху страницы при прокрутке.

Контейнер и бренд:

`<div class="container-fluid">`: создает контейнер для размещения содержимого навигационной панели.

` Тича`: определяет бренд (логотип) с текстом "Тича" и ссылкой на домашнюю страницу.

Кнопка-тогглер:

`<button class="navbar-toggler" ...>`: это кнопка-тогглер для отображения/скрытия элементов навигационной панели на мобильных устройствах.

Список навигации:

`{% if user.is_authenticated %}`: проверка, аутентифицирован ли пользователь.

Элементы списка (`<li class="nav-item"> ... `): ссылки на различные разделы приложения, такие как "Добавление записи", "Провести занятие", "Календарь событий" и т. д.

`{% else %}`: этот блок отображается, если пользователь не аутентифицирован, и содержит ссылки на "Регистрацию" и "Вход".

`span` с ID "balance": отображает текущий заработок пользователя в рублях (`{{ total_payment }}` руб).

`{% endif %}`: завершает блок условия для проверки аутентификации пользователя.

Этот шаблон предоставляет навигационную панель, которая динамически адаптируется в зависимости от статуса аутентификации пользователя, и включает различные ссылки для навигации по приложению.

5) teacha_crm\website\templates\main\history.html

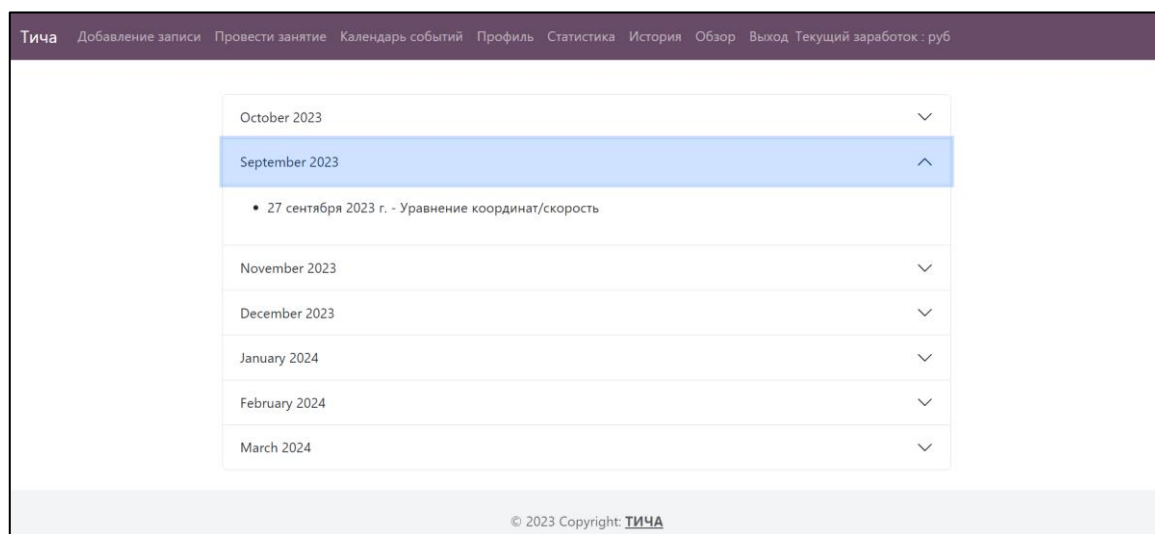


Рисунок 8 Шаблон history.html

Этот Django-шаблон создан для отображения истории уроков по месяцам. Давайте рассмотрим основные элементы:

`{% extends 'main/base.html' %}`: оповещает Django о том, что этот шаблон расширяет базовый шаблон 'main/base.html'.

`{% block content %}`: определяет блок контента, который будет вставлен в соответствующее место блока 'content' базового шаблона.

`{% load static %}`: загружает статические файлы, такие как CSS и JavaScript, используя механизм статических файлов Django.

`<div class="accordion" id="accordionExample">`: создает аккордеон для группировки и отображения уроков по месяцам.

`{% for month, lessons in lessons_by_month.items %}`: Итерирует по словарю `lessons_by_month`, который содержит уроки, сгруппированные по месяцам.

`<div class="accordion-item">`: Каждый элемент аккордеона представляет собой один месяц.

Внутри аккордеона:

`<h2 class="accordion-header">`: Заголовок, содержащий кнопку для разворачивания и сворачивания контента месяца.

`<button class="accordion-button" ...>`: Кнопка аккордеона.

`<div id="collapse{{ forloop.counter }}" ...>`: Контейнер, который может сворачиваться/разворачиваться.

`{% for lesson in lessons %}`: итерирует по урокам в текущем месяце.

`{{ lesson.lesson_date }} - {{ lesson.lesson_topic }}`: Отображает каждый урок в виде списка, содержащего дату и тему урока.

Закрывающие теги `{% endfor %}`, `{% endfor %}`, `{% endblock %}`: завершают блоки цикла и блока контента.

Этот шаблон позволяет пользователям просматривать уроки, организованные по месяцам в виде аккордеона. Каждый месяц представляет собой раскрывающийся блок, который содержит список уроков с их датой и темой.

6) teacha_crm\website\templates\main\dashboard.html

The screenshot shows a web dashboard for a CRM system. On the left is a dark sidebar with a menu: Тича, Обзор, Добавить студентов, Провести занятие, Помощь, Настройки, Профиль, Выход. The top header has a search bar and several status cards: 4 Всего участников, 7 Проведено уроков, 0.0 Доход, and 7.0 Часов в работе. The main content area is titled 'Список учеников' and contains a table with 11 columns: #, Имя Фамилия, Телефон, Стоимость, Дата создания записи, Цель, Первый день, Время первого дня, Второй день, Время второго дня, and ID. Below the table is a section for 'Ближайшие уроки'.

| # | Имя Фамилия | Телефон | Стоимость | Дата создания записи | Цель | Первый день | Время первого дня | Второй день | Время второго дня | ID |
|---|--------------------|--------------|-----------|-------------------------|-------------------------------|-------------|-------------------|-------------|-------------------|----|
| 1 | Сергей Мануковский | +79601253072 | 1000 | 5 марта 2024 г. 14:30 | Физика/ОГЭ | Вторник | 14:00 | - | - | 38 |
| 2 | Виктор Тестов | +79601253072 | 1000 | 10 ноября 2023 г. 14:05 | Физика/практика решения задач | Четверг | 15:00 | Пятница | 16:40 | 39 |
| 3 | Иван Тестов | +79601253072 | 1000 | 10 ноября 2023 г. 14:06 | Физика/ОГЭ | Понедельник | 14:00 | - | - | 40 |
| 4 | Данила Тестов | +79601253072 | 1000 | 10 ноября 2023 г. 14:07 | Физика/практика решения задач | Среда | 15:30 | - | - | 41 |

Рисунок 9 Шаблон dashboard.html

Этот Django-шаблон представляет собой веб-страницу для административной панели веб-приложения. Давайте рассмотрим основные элементы:

{% block content %}: определяет блок контента, который будет вставлен в соответствующее место блока 'content' базового шаблона.

<head>: содержит метаинформацию о веб-странице, такую как ссылки на стили и внешние ресурсы.

{% load static %}: загружает статические файлы, такие как CSS, используя механизм статических файлов Django.

<body>: определяет тело веб-страницы, содержащее основное содержание.

<div class="container">: контейнер, который ограничивает ширину содержимого страницы.

<div class="navigation">: боковая навигационная панель с пунктами меню для различных разделов приложения.

`<div class="main">`: основная часть страницы с верхней панелью, карточками статистики и деталями о пользователях.

`<div class="toggle" onclick="toggleMenu();"></div>`: кнопка переключения боковой панели.

`<div class="search">`: поле поиска.

`<div class="user">`: отображение изображения пользователя.

Карточки статистики: `total_student_records`, `total_lesson_records`, `total_payment`, `total_hours`: переменные, содержащие статистическую информацию. Каждая карточка имеет соответствующее изображение и название.

Таблица с данными о пользователях (`records`): содержит информацию о каждом пользователе в виде строк таблицы. Каждая строка содержит информацию, такую как имя, фамилия, телефон, стоимость и дата создания записи.

Скрипт JavaScript в конце для переключения боковой панели.

Использование иконок Font Awesome для стилизации иконок в меню и карточках.

Шаблоны, содержащиеся в директории `teacha_crm\website\templates\profile`

```
teacha_crm\website\templates\profile\register.html
teacha_crm\website\templates\profile\profile.html
teacha_crm\website\templates\profile\edit_profile.html
teacha_crm\website\templates\profile\delete_profile.html
teacha_crm\website\templates\profile\statistic.html
```

1) `teacha_crm\website\templates\profile\register.html`

Этот Django-шаблон представляет собой страницу регистрации. Рассмотрим его основные элементы:

`{% extends 'main/base.html' %}`: этот шаблон расширяет базовый шаблон 'main/base.html', что означает, что он включает в себя все содержимое базового шаблона, но заменяет блок content содержимым, указанным в этом шаблоне.

`{% load static %}`: загружает статические файлы, такие как CSS, используя механизм статических файлов Django.

ТИЧА Регистрация Вход

Регистрация

Имя пользователя
Требуемый: 150 символов или меньше. Только буквы, цифры и @/./+/-/_

Ваше имя

Ваша фамилия

Электронная почта

Пароль

- Ваш пароль не должен быть слишком похож на другую вашу личную информацию.
- Ваш пароль должен содержать не менее 8 символов.
- Ваш пароль не может быть часто используемым паролем.
- Ваш пароль не может быть полностью цифровым.

Подтвердите пароль
Повторите ввод пароля

Регистрация в ТИЧА

© 2023 Copyright: ТИЧА

Рисунок 10 Шаблон register.html

`{% block content %}`: определяет блок контента, который будет вставлен в соответствующее место блока content базового шаблона.

`<div class="col-md-6 offset-md-3">`: создает контейнер с шириной 6 колонок и смещением в 3 колонки от левого края.

`<h1>Регистрация</h1>`: заголовок страницы "Регистрация".

`<form method="POST" action="">`: определяет HTML-форму с методом POST для отправки данных на сервер.

`{% csrf_token %}`: вставляет защитный токен CSRF для обеспечения безопасности формы.

`{% if form.errors %}...{% endif %}`: проверяет наличие ошибок валидации формы и выводит их, если они есть.

`{{ form.as_p }}`: отображает форму в виде абзацев.

`<button type="submit" class="btn btn-secondary">Регистрация в ТИЧА</button>`: кнопка отправки формы.

`</div>`: закрытие контейнера.

`{% endblock %}`: завершение блока контента.

2) `teacha_crm\website\templates\profile\profile.html`

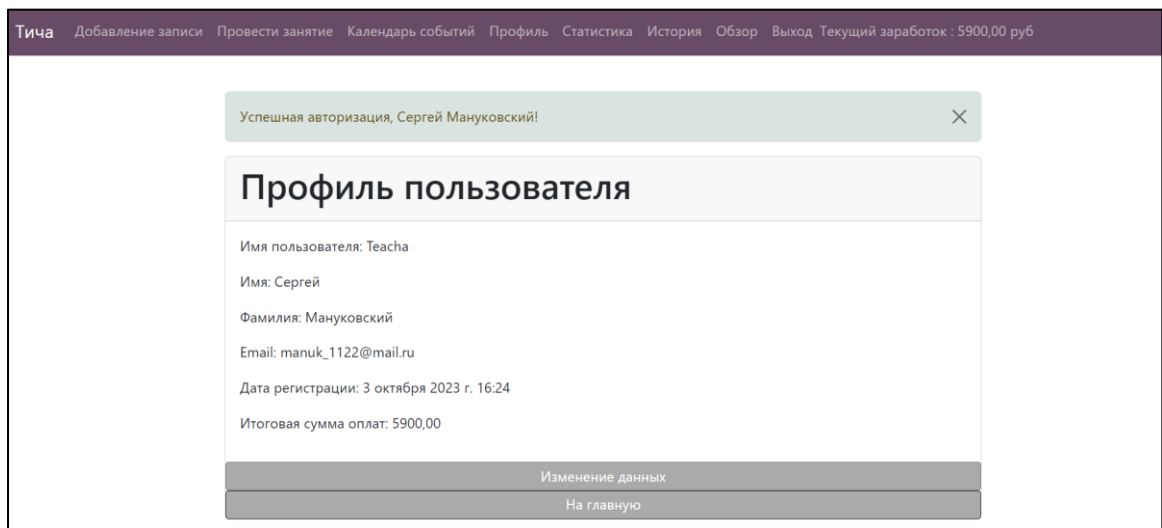


Рисунок 11 Шаблон `profile.html`

Этот Django-шаблон представляет собой страницу профиля пользователя. Давайте рассмотрим основные элементы:

`{% extends 'main/base.html' %}`: указывает, что этот шаблон расширяет базовый шаблон `'main/base.html'`.

`{% load static %}`: загружает статические файлы, такие как CSS, с использованием механизма статических файлов Django.

`{% block content %}`: определяет блок контента, который будет вставлен в соответствующее место блока `'content'` базового шаблона.

`<div class="card">`: создает блок карточки для размещения информации о профиле пользователя.

`<div class="card-header">`: заголовок карточки с названием "Профиль пользователя".

`<div class="card-body">`: тело карточки с информацией о пользователе и итоговой сумме оплаты.

Переменные `{{ user.username }}`, `{{ user.first_name }}`, `{{ user.last_name }}`, `{{ user.email }}`, `{{ user.date_joined }}`, `{{ total_payment }}`: отображают информацию о пользователе и итоговую сумму оплаты.

Ссылки для редактирования профиля и возврата на главную страницу:

``Изменение данных``: ссылка на страницу редактирования профиля.

``На главную``: ссылка на главную страницу.

3) teacha_crm\website\templates\profile\edit_profile.html

Рисунок 12 Шаблон edit_profile.html

Этот Django-шаблон представляет собой страницу для обновления записи (профиля пользователя). Давайте рассмотрим основные элементы:

`{% extends 'main/base.html' %}`: указывает, что этот шаблон расширяет базовый шаблон 'main/base.html'.

`{% load static %}`: загружает статические файлы, такие как CSS, с использованием механизма статических файлов Django.

`{% block content %}`: определяет блок контента, который будет вставлен в соответствующее место блока 'content' базового шаблона.

`<div class="col-md-6 offset-md-3">`: создает контейнер с использованием сеточной системы Bootstrap, занимающий 6 колонок посередине ширины страницы.

`<h1>Обновление записи</h1>`: заголовок страницы, указывающий на то, что это страница обновления записи.

`<form method="POST">`: открывает форму для отправки данных методом POST.

`{% csrf_token %}`: вставляет метку CSRF-токена для безопасной обработки формы.

`{{ form.as_p }}`: Отображает форму в виде абзацев.

`<div class="btn-group">`: создает группу кнопок для навигации и взаимодействия с записью.

Ссылки:

`На главную`: Ссылка для возврата на главную страницу.

`<button type="submit" class="btn btn-secondary">Обновить запись</button>`: Кнопка для отправки формы и обновления записи.

`Удалить
профиль`: Ссылка для удаления профиля.

`</form>`: Закрывает тег формы.

Шаблон предоставляет пользователю форму для внесения изменений в запись и взаимодействия с ней. Он также содержит ссылки для навигации по другим страницам.

4) `teacha_crm\website\templates\profile\delete_profile.html`

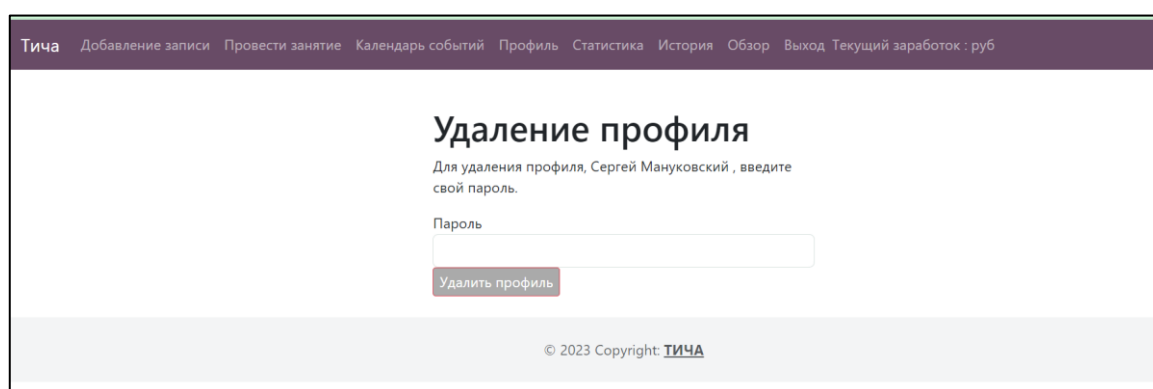


Рисунок 13 Шаблон `delete_profile.html`

Этот Django-шаблон представляет собой страницу для удаления профиля пользователя. Давайте разберем основные элементы:

`{% extends 'main/base.html' %}`: Указывает, что этот шаблон расширяет базовый шаблон `'main/base.html'`.

`{% load static %}`: загружает статические файлы, такие как CSS, с использованием механизма статических файлов Django.

`{% block content %}`: определяет блок контента, который будет вставлен в соответствующее место блока `'content'` базового шаблона.

`<div class="col-md-6 offset-md-3">`: создает контейнер с использованием сеточной системы Bootstrap, занимающий 6 колонок посередине ширины страницы.

`<h1>Удаление профиля</h1>`: заголовок страницы, указывающий на то, что это страница удаления профиля.

`<p>`Для удаления профиля, `{{ user.first_name }}` `{{ user.last_name }}`, введите свой пароль.`</p>`: Текстовое описание, инструктирующее пользователя ввести свой пароль для удаления профиля.

`<form method="POST">`: открывает форму для отправки данных методом POST.

`{% csrf_token %}`: вставляет метку CSRF-токена для безопасной обработки формы.

`<div class="form-group">`: Группа формы для стилизации элементов формы.

`<label for="password">`Пароль`</label>`: Метка для поля ввода пароля.

`<input type="password" class="form-control" id="password" name="password" required />`: поле ввода пароля с типом "password" и обязательным заполнением.

`<button type="submit" class="btn btn-danger">`Удалить профиль`</button>`: Кнопка для отправки формы и удаления профиля.

`</form>`: закрывает тег формы.

Шаблон предоставляет пользователю форму для ввода пароля и последующего удаления профиля.

5) `teacha_crm\website\templates\profile\statistic.html`

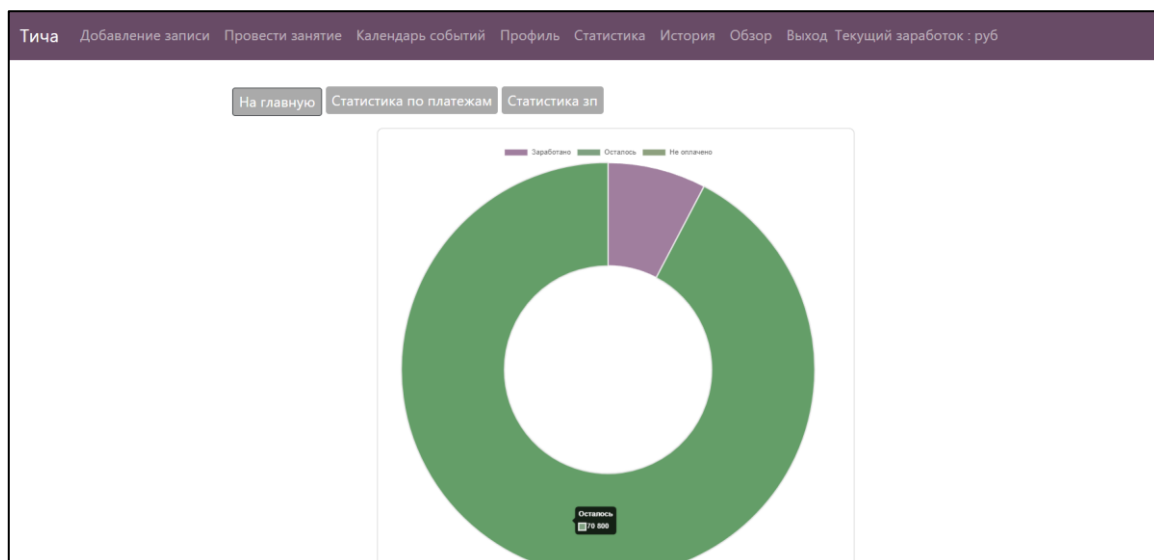


Рисунок 14 Шаблон statistic.html

Этот Django-шаблон представляет собой страницу статистики с использованием библиотеки Chart.js. Давайте разберем основные элементы:

`{% extends 'main/base.html' %}`: указывает, что этот шаблон расширяет базовый шаблон 'main/base.html'.

`{% load static %}`: загружает статические файлы, такие как CSS, с использованием механизма статических файлов Django.

`<link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}" />`: Подключает стили из файла 'css/styles.css'.

`<title>Статистика</title>`: устанавливает заголовок страницы.

`На главную`: кнопка для перехода на главную страницу.

Секция с кнопками "Статистика по платежам" и "Статистика зп".

`<div id="chart-container">`: контейнер для вставки графиков.

`<div class="collapse collapse-horizontal" id="collapse_gonorar">`: сворачиваемый блок для статистики по платежам.

`<canvas id="myChart"></canvas>`: элемент Canvas для рендеринга графика с использованием Chart.js.

`<div class="collapse collapse-horizontal" id="collapse_day">`: сворачиваемый блок для статистики зп.

`<canvas id="myPieChart"></canvas>`: элемент Canvas для рендеринга круговой диаграммы с использованием Chart.js.

`<script>...</script>`: JavaScript-код для инициализации и настройки графиков с использованием библиотеки Chart.js.

График статистики по платежам (гистограмма).

Круговая диаграмма статистики зп.

Перед JavaScript-кодом есть переменные `student_names` и `total_payments`, которые передаются в шаблон из Django-контекста. Эти переменные используются для построения данных графика.

Шаблоны, содержащиеся в директории
teacha_crm\website\templates\student

| |
|---|
| teacha_crm\website\templates\student\add_record.html |
| teacha_crm\website\templates\student\record.html |
| teacha_crm\website\templates\student\update_record.html |

1) teacha_crm\website\templates\student\add_record.html

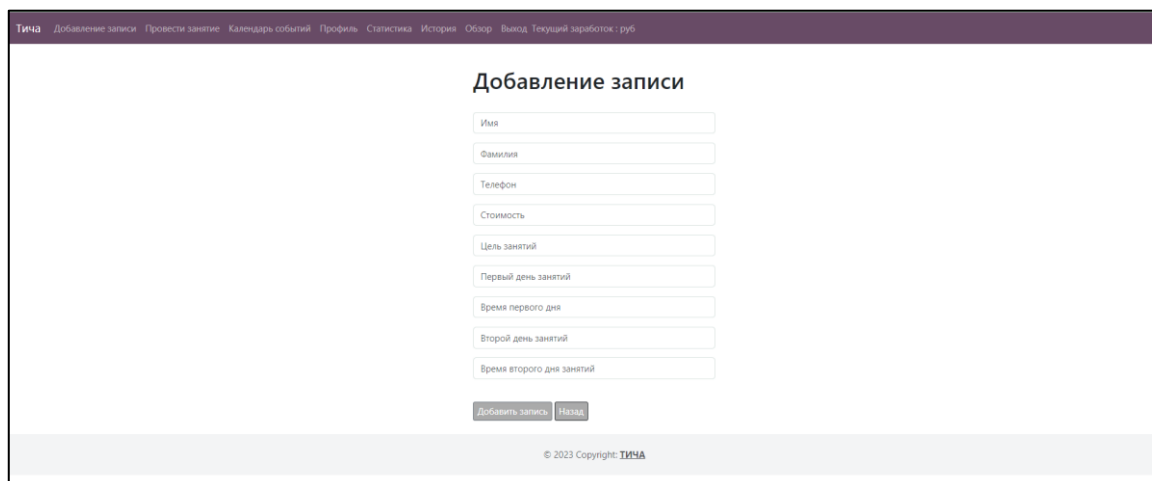


Рисунок 15 Шаблон add_record.html

Этот Django-шаблон представляет собой страницу добавления записи с использованием формы. Давайте разберем основные элементы:

`{% extends 'main/base.html' %}`: указывает, что этот шаблон расширяет базовый шаблон 'main/base.html'.

`{% load static %}`: загружает статические файлы, такие как CSS, с использованием механизма статических файлов Django.

`<link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}" />`: подключает стили из файла 'css/styles.css'.

`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.css" />`: подключает стили из библиотеки Font Awesome для использования значков.

`<div class="col-md-6 offset-md-3">`: определяет контейнер для центрирования содержимого формы в середине страницы.

`<h1>Добавление записи</h1>`: заголовок страницы.

`<form method="POST" action="{% url 'add_record' %}">`: форма для ввода данных записи. Метод POST используется для отправки данных, и action указывает URL, куда отправлять данные формы.

{% csrf_token %}: вставляет токен CSRF для защиты от атаки подделки межсайтовых запросов.

{{ form.as_p }}: вставляет HTML-представление формы в виде абзацев.

<button type="submit" class="btn btn-secondary">Добавить запись</button>:
Кнопка для отправки формы.

Назад: кнопка для возврата на главную страницу.

{% endblock %}: завершает блок контента страницы.

Этот шаблон предназначен для страницы добавления записи, и он использует стили из файла 'css/styles.css', а также значки из библиотеки Font Awesome.

2) teacha_crm\website\templates\student\record.html

Этот Django-шаблон представляет собой страницу просмотра деталей записи студента. Рассмотрим основные элементы:

{% extends 'main/base.html' %}: указывает, что этот шаблон расширяет базовый шаблон 'main/base.html'.

{% block content %}: начало блока контента страницы.

Тича Добавление записи Провести занятие Календарь событий Профиль Статистика История Обзор Выход Текущий заработок : руб

Имя
Сергей Мануковский

Телефон: +79601253072
Дата созданий записи: 5 марта 2024 г. 14:30

Уроки студента

| # | Дата урока | Продолжительность | Тема | Оплата | Статус |
|---|------------------|-------------------|---------------------------------------|---------|-------------|
| 1 | 2 ноября 2023 г. | 01:30:00 | Решение задач по теплоемкости | 1000.00 | Оплачено |
| 2 | 9 ноября 2023 г. | 01:30:00 | Решение задач на силу тренияфвфвфвфвф | 1000.00 | Не оплачено |
| 3 | 5 ноября 2023 г. | 01:00:00 | Решение задач по теплоемкости | 1000.00 | Оплачено |

На главную Назад Обновить запись Удалить Провести занятие

Рисунок 16 Шаблон record.html

{% load static %}: загружает статические файлы, такие как CSS и JavaScript.

`<link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}" />`: подключает стили из файла 'css/styles.css'.

`<div class="card">`: создает блок карты для отображения основных данных студента.

`<div class="card-header">`: заголовок карты, отображающий имя и фамилию студента.

`<div class="card-body">`: тело карты, содержащее основные детали студента, такие как телефон и дата создания записи.

`<h2>Уроки студента</h2>`: заголовок, отображающий список уроков студента.

`<div class="table-container">`: контейнер для таблицы, содержащей информацию о уроках студента.

`<table class="table table-striped table-hover table-sm">`: таблица для отображения уроков студента.

`{% for lesson in student_lessons %}`: цикл для итерации по списку уроков студента.

`{{ lesson.lesson_topic }}`: ссылка на страницу записи урока с отображением темы урока.

`На главную`: кнопка для перехода на главную страницу.

`Назад`: кнопка для возврата на предыдущую страницу в истории браузера.

`Обновить запись`: кнопка для перехода к странице обновления записи студента.

`Удалить`: кнопка для удаления записи студента с использованием JavaScript для подтверждения действия.

`Провести занятие`: кнопка для перехода к странице добавления урока.

`<script src="{% static 'scripts/confirmDelete.js' %}"></script>`: подключает скрипт 'scripts/confirmDelete.js' для подтверждения удаления записи.

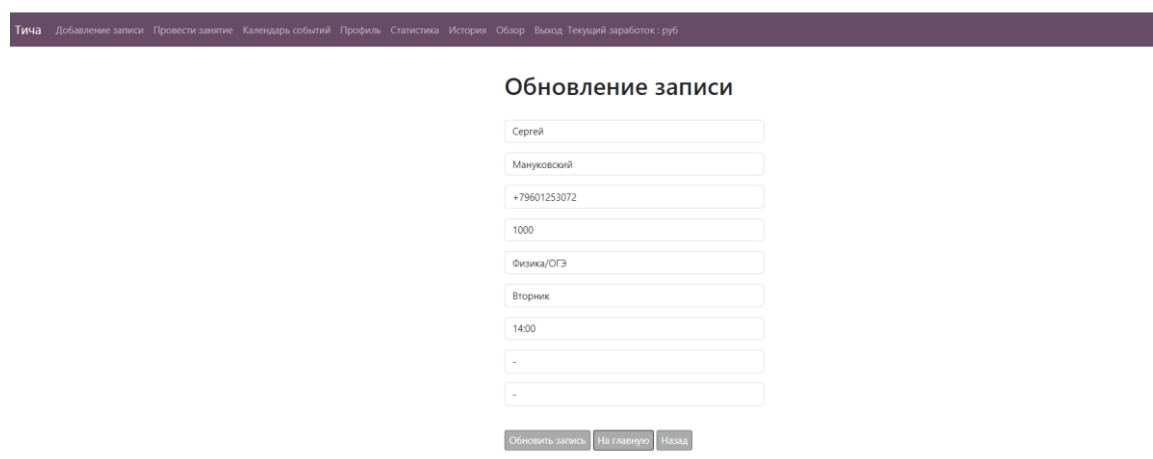
`{% endblock %}`: завершение блока контента страницы.

Этот шаблон предназначен для отображения деталей студента, включая его уроки, и предоставляет кнопки для взаимодействия с записью.

3) `teacha_crm\website\templates\student\update_record.html`

Этот HTML-шаблон представляет страницу для обновления записи. Давайте разберем основные элементы:

`{% extends 'main/base.html' %}`: этот тег указывает, что этот шаблон расширяет (inherit) другой шаблон, в данном случае, 'main/base.html'.



Тича | Добавление записи | Провести занятие | Календарь событий | Профиль | Статистика | История | Обзор | Выход | Текущий заработок: руб

Обновление записи

Рисунок 17 Шаблон `update_record.html`

`{% block content %}`: открывается блок с именем "content". Это место, куда будет вставлен контент, который предоставляется в дочернем шаблоне.

`<div class="col-md-6 offset-md-3">`: определяет контейнер для центрирования содержимого формы в середине страницы.

`<h1>Обновление записи</h1>`: Заголовок страницы.

`<form method="POST">`: форма для ввода данных записи. Метод POST используется для отправки данных.

`{% csrf_token %}`: вставляет токен CSRF для защиты от атаки подделки межсайтовых запросов.

`{{ form.as_p }}`: вставляет HTML-представление формы в виде абзацев.

`<button type="submit" class="btn btn-secondary">Обновить запись</button>`: кнопка для отправки формы и обновления записи.

`На главную`: кнопка для перехода на главную страницу.

`Назад`: кнопка для возврата на предыдущую страницу в истории браузера.

`{% endblock %}`: завершает блок контента страницы.

Этот шаблон позволяет пользователям обновлять записи, вводя новые данные и отправляя форму.

Шаблоны, содержащиеся в директории
teacha_crm\website\templates\lesson

| |
|---|
| teacha_crm\website\templates\lesson\add_lessons.html teacha_crm\website\templates\lesson\lesson_record.html teacha_crm\website\templates\lesson\confirm_delete_lesson.html teacha_crm\website\templates\lesson\calendar.html |
|---|

1) teacha_crm\website\templates\lesson\add_lessons.html

Этот HTML-шаблон представляет страницу для добавления урока. Давайте разберем основные элементы:

`{% extends 'main/base.html' %}`: этот тег указывает, что этот шаблон расширяет (inherit) другой шаблон, в данном случае, 'main/base.html'.

`{% block content %}`: открывается блок с именем "content". Это место, куда будет вставлен контент, который предоставляется в дочернем шаблоне.

`<div class="col-md-6 offset-md-3">`: определяет контейнер для центрирования содержимого формы в середине страницы.

`<h1>Добавление Урока</h1>`: заголовок страницы.

`<form method="POST" action="{% url 'add_lessons' %}">`: форма для ввода данных урока. Метод POST используется для отправки данных, и action указывает на URL, куда отправлять данные.

`{% csrf_token %}`: вставляет токен CSRF для защиты от атаки подделки межсайтовых запросов.

`{{ form.as_p }}`: вставляет HTML-представление формы в виде абзацев.

`<div class="btn-group">`: группа кнопок.

`<button type="submit" class="btn btn-secondary">Добавить запись</button>`: кнопка для отправки формы и добавления урока.

`Назад`: кнопка для перехода на главную страницу.

`{% endblock %}`: завершает блок контента страницы.

Этот шаблон позволяет пользователям добавлять новые уроки, вводя необходимые данные и отправляя форму.

Рисунок 18 Шаблон add_lessons.html

2) teacha_crm\website\templates\lesson\lesson_record.html

Шаблон lesson_record.html представляет собой страницу с информацией об уроке и возможностью его редактирования. Давайте разберем каждый элемент:

{% extends 'main/base.html' %}: этот тег указывает, что этот шаблон расширяет (наследует) другой шаблон, в данном случае, 'main/base.html'. Это общий подход, когда вы хотите использовать один основной макет для нескольких страниц.

{% block content %}: открывается блок с именем "content". Это место, где будет вставлен контент, предоставляемый в дочернем шаблоне.

Рисунок 19 Шаблон lesson_record

`<h1>Информация об уроке</h1>`: заголовок страницы, который говорит о том, что эта страница содержит информацию об уроке.

`<p>Дата урока: {{ lesson.lesson_date }}</p>`: выводит дату урока.

`<p>Продолжительность: {{ lesson.lesson_duration }}</p>`: выводит продолжительность урока.

`<p>Тема: {{ lesson.lesson_topic }}</p>`: выводит тему урока.

`<div class="card"> ... </div>`: создается блок карты для отображения комментария на следующее занятие.

`<div class="card-header"> ... </div>`: заголовок блока карты, который содержит "Комментарий на следующее занятие:".

`{{ lesson.next_lesson_comment }}`: выводит текст комментария на следующее занятие.

`<p>Оплата: {{ lesson.payment }}</p>`: выводит информацию об оплате за урок.

`<h2>Редактировать урок</h2>`: заголовок, который говорит о возможности редактирования урока.

`<form method="post"> ... </form>`: форма для отправки данных при редактировании урока. Использует метод POST для отправки данных.

`{% csrf_token %}`: вставляет токен CSRF для защиты от атаки подделки межсайтовых запросов.

`{{ form.as_p }}`: вставляет HTML-представление формы в виде абзацев.

`На главную`: кнопка для перехода на главную страницу.

`Назад`: кнопка для возвращения на предыдущую страницу в истории браузера.

`<button class="btn btn-secondary" type="submit">Сохранить</button>`: кнопка для отправки формы и сохранения редактированных данных урока.

`Удалить`: кнопка для удаления урока.

`{% endblock %}`: закрывает блок контента страницы.

Этот шаблон предоставляет удобный интерфейс для просмотра информации об уроке, редактирования и удаления урока.

3) `teacha_crm\website\templates\lesson\confirm_delete_lesson.html`

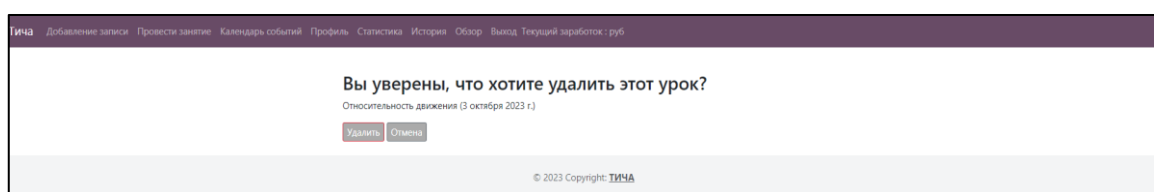


Рисунок 20 Шаблон `confirm_delete_lesson.html`

Этот HTML-шаблон представляет страницу подтверждения удаления урока. Рассмотрим основные элементы:

`{% extends 'main/base.html' %}`: этот тег указывает, что этот шаблон расширяет другой шаблон, 'main/base.html'.

`{% block content %}`: открывается блок с именем "content". Это место, куда будет вставлен контент, предоставляемый в дочернем шаблоне.

`<h2>Вы уверены, что хотите удалить этот урок?</h2>`: заголовок страницы с вопросом о подтверждении удаления.

`<p>{{ lesson.lesson_topic }} ({{ lesson.lesson_date }})</p>`: отображение информации об уроке, который собираются удалить.

`<form method="post">`: форма для отправки данных. Метод POST используется для отправки данных, и это обычно используется для удаления ресурса.

`{% csrf_token %}`: вставляет токен CSRF для защиты от атаки подделки межсайтовых запросов.

`<button type="submit" class="btn btn-danger">Удалить</button>`: кнопка для отправки формы и подтверждения удаления урока. Стилизована как красная кнопка для обозначения действия удаления.

`Отмена`: кнопка для отмены операции удаления и возврата на главную страницу. Стилизована как серая кнопка.

4) `teacha_crm\website\templates\lesson\calendar.htm`

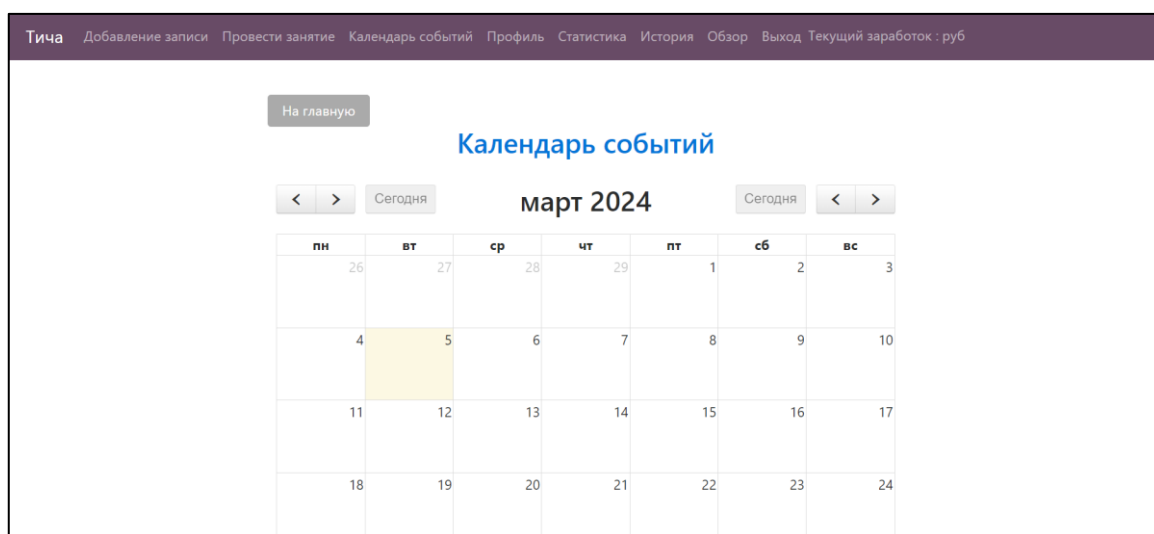


Рисунок 21 Шаблон `calendar.html`

Этот HTML-шаблон представляет страницу календаря событий. Давайте рассмотрим основные элементы:

`{% extends 'main/base.html' %}`: этот тег указывает, что этот шаблон расширяет другой шаблон, 'main/base.html'.

`{% block content %}`: открывается блок с именем "content". Это место, куда будет вставлен контент, предоставляемый в дочернем шаблоне.

`<html lang="ru">`: тег для установки языка страницы на русский.

`<head>`: секция с метаинформацией, стилями и скриптами.

`На главную`: Кнопка для перехода на главную страницу.

Загрузка стилей и библиотек, таких как FullCalendar, Bootstrap, и других.

JavaScript-скрипты для работы с календарем и библиотекой Toastr.

<body>: открывается тело документа.

<h2 align="center">Календарь событий</h2>: заголовок страницы.

<div class="container">: контейнер для размещения календаря.

<div id="calendar"></div>: элемент для отображения календаря.

{% endblock %}: завершает блок контента страницы.

Этот шаблон использует библиотеку FullCalendar и библиотеку Toastr для создания интерактивного календаря событий. Календарь позволяет добавлять, редактировать и удалять события с помощью AJAX-запросов.

Хранение статических файлов (CSS, JavaScript, изображения) в папке teacha_crm\static приложения.

Запуск локального сервера разработки с помощью команды python manage.py runserver для проверки и отладки приложения. Так же доступ к сервису организован по <http://185.180.109.118//>. Для тестового входа использовать следующие учетные данные:

Логин: Test

Пароль : ntcnjdsqgjkmpjdfntkm

Порядок деплоя сайта на сервер следующий:

- git clone https://github.com/BackDad/TEACHA_CRM.git: Клонирование репозитория TEACHA_CRM с GitHub.
- cd TEACHA_CRM/: Переходит в каталог TEACHA_CRM.
- python3 -m venv myenv: Создает виртуальное окружение с именем myenv.
- source myenv/bin/activate: Активирует виртуальное окружение.

- `pip install -r requirements.txt`: Устанавливает зависимости из файла `requirements.txt`.

- `pip install gunicorn`: Устанавливает Gunicorn.
- `vim teacha_crm/teacha_crm/settings.py`: Открывает файл настроек проекта в редакторе Vim для редактирования. (указывается адрес сервера)
- `vim /etc/systemd/system/teacha_crm.gunicorn.service`: Открывает файл службы Gunicorn в редакторе Vim для редактирования.
- `systemctl enable teacha_crm.gunicorn`: Включает автозапуск службы Gunicorn при загрузке системы.
- `systemctl start teacha_crm.gunicorn`: Запускает службу Gunicorn.

3.2 Дальнейшее развитие CRM системы ТИЧА: Инновационные шаги в образовании

Современное образование стремительно эволюционирует, и эффективное управление учебными процессами становится ключевым фактором успеха. В этой статье мы рассмотрим несколько ключевых направлений развития CRM системы ТИЧА и представим идеи для инноваций, которые могут значительно улучшить пользовательский опыт и результаты обучения.

- **Добавление системы оплаты:** Интеграция онлайн-платежей позволит репетиторам принимать оплату за занятия напрямую через платформу, обеспечивая удобство для всех сторон.
- **Стилизация под конкретный предмет:** Предоставление возможности выбора тематических стилей для учебных материалов и интерфейса позволит создавать персонализированные образовательные среды.
- **Добавление групповых занятий:** Расширение функционала для проведения групповых занятий позволит репетиторам эффективно управлять группами учеников и проводить коллективные занятия.
- **Добавление личного кабинета учащегося:** Создание личного кабинета для учеников позволит им управлять своим расписанием, просматривать материалы и отслеживать свой прогресс.
- **Добавление новых видов статистики:** Внедрение расширенной аналитики, включающей оценку успеваемости, динамику прогресса и анализ эффективности занятий, поможет репетиторам и ученикам выявлять сильные и слабые стороны и принимать обоснованные решения.
- **Интеграция с госуслугами:** Возможность автоматической передачи данных об учениках в государственные информационные системы образования упростит процессы отчетности и взаимодействия с образовательными учреждениями.

- Восстановление пароля: Добавление функционала восстановления пароля через электронную почту или SMS обеспечит безопасность и удобство доступа к системе.
- Коммуникация с учащимися: Интеграция чата или системы обмена сообщениями позволит репетиторам и ученикам эффективно общаться в режиме реального времени.
- Хранение файлов домашней работы: Создание центра загрузки и хранения файлов позволит ученикам и репетиторам обмениваться материалами для занятий и домашними заданиями.
- Обратная связь с родителями: Предоставление возможности родителям отслеживать прогресс и результаты обучения своих детей через специальный портал или приложение.
- Возможность передавать уроки другим репетиторам (Биржа уроков): Создание платформы для обмена уроками между репетиторами позволит оптимизировать расписание и обеспечить доступность обучения в различных форматах.
- Общение между репетиторами: Внедрение системы обмена опытом и знаниями между репетиторами способствует профессиональному развитию и совместному решению задач.
- Форум: Создание образовательного форума для обсуждения вопросов и обмена опытом между учениками, родителями и репетиторами.
- Дальнейшее развитие CRM системы ТИЧА направлено на создание универсальной образовательной платформы, обеспечивающей удобство использования, эффективное управление образовательными процессами и поддержку взаимодействия между всеми участниками образовательного процесса.

Заключение

Процесс разработки CRM системы ТИЧА был крайне интенсивным и продуктивным. За время работы над проектом было затрачено около 400 часов интенсивной трудовой деятельности. Этот период был насыщен глубоким анализом требований, проектированием функционала и активным программированием.

В ходе работы было создано более 5000 строк кода, что подчеркивает масштабность и сложность разрабатываемой системы. Каждая строка кода была тщательно проработана с целью обеспечения эффективности, безопасности и удобства пользования.

Проведено исследование 15 книг по темам Django, Linux, HTML, CSS, JS, Mysql, Nginx, Gunicorn, Python, Bootstrap. Этот этап позволил углубить понимание технологических аспектов проекта и внедрить передовые практики разработки.

База данных об уроках собрана и составляет 296 записей. Это ценный ресурс для последующего анализа, оптимизации процессов и дальнейшего улучшения системы в ответ на потребности пользователей.

Важным аспектом проекта стали затраты, оценивающиеся в 30 тысяч рублей. Эти средства были направлены на оплату хостинга, приобретение необходимых технологических ресурсов и поддержание проекта в активном состоянии.

В целом, разработка CRM системы ТИЧА стала значимым этапом в области онлайн-образования, предоставив участникам образовательного процесса инновационные инструменты управления и взаимодействия. Реализованные функциональности и перспективные направления развития делают проект перспективным и востребованным среди репетиторов и учеников.

Список используемой литературы

- 1) "Django для начинающих" от Уильяма С. Винсента
- 2) "Two Scoops of Django 3.x: Лучшие практики для веб-фреймворка Django" от Даниэля Роя Гринфельда и Одри Рой Гринфельд
- 3) "Командная строка Linux" от Уильяма Шоттса младшего
- 4) "Библия Linux" от Кристофера Негуса
- 5) "HTML и CSS: Создание веб-сайтов" от Джона Дакетта
- 6) "JavaScript: Хорошие части" от Дугласа Крокфорда
- 7) "Изучаем MySQL" от Робина Никсона
- 8) "Высокопроизводительный MySQL: Оптимизация, резервное копирование и репликация" от Барона Шварца, Петера Зайтсева, Вадима Ткаченко и Джереми Заводни
- 9) "Сервер HTTP Nginx - Третье издание" от Клемента Неделку
- 10) "Освоение Nginx" от Димитри Айвалиотиса
- 11) Официальная документация (<https://docs.gunicorn.org/>)
- 12) "Развертывание Django с использованием Gunicorn и Nginx" (<https://testdriven.io/blog/dockerizing-django-with-postgres-gunicorn-and-nginx/>)
- 13) "Python как профессионал" от Лучано Рамальо
- 14) "Python. Курс программирования" от Эрика Мэттиса
- 15) "Быстрый старт с Bootstrap 4: Разработка отзывчивого веб-дизайна и разработка с Bootstrap 4" от Джейкоба Летта