МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

Образовательное учреждение высшего образования

«Челябинский государственный университет»

(ФГБОУ ВО «ЧелГУ»)

Колледж ЧелГУ

09.02.04 Информационные системы (по отраслям)

КУРСОВОЙ ПРОЕКТ

По дисциплине: Методы и средства проектирования

На тему: **Разработка информационной системы для автоматического проектирования расписания в колледже и проектирование АРМ сотрудника УМР**

Выполнил: студент

Группы СПМИС-301

Артемий Кириллович Киселев

Руководитель:

Кирилл Андреевич Лихачёв

Оценка\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_

Челябинск

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc135054052)

[1. Теоретическая часть 4](#_Toc135054053)

[1.1 Анализ требований и функциональности информационной системы для автоматического проектирования расписания 4](#_Toc135054054)

[1.2 Выбор технологического стека 5](#_Toc135054055)

[2. Практическая часть 9](#_Toc135054056)

[2.1 Техническое задание 9](#_Toc135054057)

[2.2 Установка программ для работы, создание Django проекта 10](#_Toc135054058)

[2.3 Создание и подключение базы данных, настройка моделей 14](#_Toc135054059)

[2.4 Написание алгоритма генерации 18](#_Toc135054060)

[ЗАКЛЮЧЕНИЕ 24](#_Toc135054062)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc135054063)

[Приложение 1 26](#_Toc135054064)

[Приложение 2 30](#_Toc135054065)

[Приложение 3 30](#_Toc135054065)

ВВЕДЕНИЕ

В эпоху цифровой трансформации, когда даже самые обычные задачи становятся автоматизированными, появляется все больше потребностей в различных веб-сервисах. Один из таких сервисов - это веб-сайт для генерации расписания. Будь то расписание для школьников, студентов, сотрудников компании или даже тренажерного зала, цифровой подход значительно упрощает процесс управления временем и ресурсами.

Создание веб-сайта для генерации расписания – это многоуровневый процесс, который включает в себя понимание требований пользователей, проектирование интерфейса, разработку логики функционирования и поддержку в рабочем состоянии. Для эффективной реализации проекта необходимо не только технические навыки, но и глубокое понимание бизнес-процессов, к которым прикладывается система.

В рамках этой задачи мы подробно рассмотрим все аспекты создания веб-сервиса для генерации расписания, начиная от анализа требований и заканчивая этапами внедрения и поддержки. Мы исследуем важные темы, такие как выбор технологического стека, определение функциональности, разработка дизайна и интерфейса.

Целью данного изложения является предоставление полного и понятного представления о процессе создания веб-сайта для генерации расписания, чтобы помочь начинающим разработчикам и другим заинтересованным лицам успешно реализовать подобные проекты.

1. Теоретическая часть
   1. Анализ требований и функциональности информационной системы для автоматического проектирования расписания

Сервис должен быть способен обрабатывать данные о различных группах и соответствующем количестве пар в неделю. Это подразумевает необходимость разработки алгоритма, способного оптимально распределить занятия на протяжении недели, с учетом того, что преподаватель может вести несколько предметов в разных группах, а также предметы могут занимать одинаковые кабинеты.

Важной функциональностью сервиса будет возможность добавления, удаления и редактирования групп, предметов, учителей и кабинетов их нагрузки в базе данных, что обеспечит гибкость и актуальность информации. Также, сервис должен обеспечивать генерацию расписания в удобном для пользователя формате, с возможностью его просмотра на сайте.

Основываясь на этих данных, можно определить первоначальные требования к функциональности сайта, которые послужат основой для дальнейшего проектирования и разработки.

Интерфейс расписания должен быть основан на существующем расписании колледжа «ЧелГУ» и иметь похожий и понятный интерфейс, чтобы не вызывать проблем у пользователя к его восприятию (Рисунок 1)

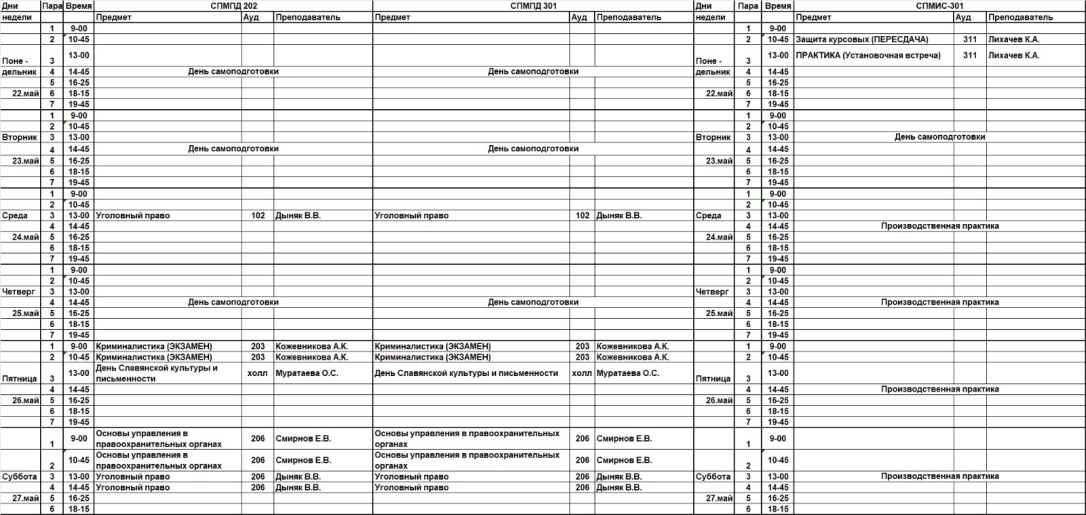


Рисунок 1 - пример расписания колледжа ЧелГУ

Важно также учитывать, что у преподавателя не может быть 2 предмета в разных группах одновременно, а также преподаватель не может иметь больше, чем n-ое количество предметов в день, и чтобы предметы распределялись принципом наибольшей полезности (чтобы преподавателю не нужно было приходить, например, на 1 предмет посередине дня)

* 1. Выбор технологического стека

В качестве технологического стека для серверной части мы можем использовать несколько вариантов:

1. Flask
2. Fast Api
3. Django
4. GoLang RPC
5. Flask – довольно простой фреймворк, использующий шаблонизатор Jinja 2 для рендеринга html страниц.   
   Плюсы: Достаточно простой для изучения, подходит для небольших приложений

Минусы: Очень мало функций из коробки, для установки дополнительных потребуется их поискать, и почитать что они делают. Нет встроенной ORM, придется также подключать SQLAlchemy и самостоятельно подключаться к Базе данных.

1. Fast Api – асинхронный фреймворк для написания API серверного приложения.

Плюсы: Асинхронный, подходит для написания быстрого API для небольших или даже высоконагруженных систем.

Минусы: Нет поддержки рендеринга страниц, что означает нам придется использовать JavaScript для динамической отрисовки страницы по данным с сервера. Нет ORM из коробки, также придется подключать SQLAlchemy и самостоятельно делать подключения к базе данных.

1. Django – отличный веб фреймворк для написания веб приложений любой сложности. Поддерживает как синхронную, так и ассинхронную версию

Плюсы: Легко настроить и подключить приложения, есть хорошая библиотека DRF (Django REST Framework) для написания API любой сложности. Есть встроенная ORM и отрисовщик шаблонов. Отрисовщик шаблонов позволяет принимать даже ORM модели. Легко пишутся модели для базы данных

Минусы: Довольно много функций из коробки, много функций может не понадобиться. Порог вхождения выше, чем у предыдущих фреймворков.

1. GolangRPC – библиотека, написанная на языке программирования Go. Позволяет сделать быстрый асинхронный простой веб-сервер. Сам язык Go очень минимизирован и построен на архитектуре языка C, а также выполняется асинхронно.

Плюсы: Идеально подходит для написания микросервисной архитектуры. Также идеально подходит для написания совсем небольшого API приложения. Выполняет асинхронные запросы в Базу Данных в рамках одной транзакции, что ускоряет различные запросы в БД.

Минусы: не поддерживает рендер html документов из коробки, есть пользовательская, но очень неудобная в использовании ORM (gorm). Требуется понимание асинхронной архитектуры приложения и самого устройства языка программирования Go.

На основе этой информации я решил выбрать связку Django + Django REST Framework. Оно идеально подходит под задачу: можно легко сделать модели кабинетов, групп, учителей и предметов в базу данных, легко фильтровать модели в базе данных. Также можно сделать как полноценный веб-сервер с динамическими html страницами за счет рендера, встроенного в сам фреймворк, так и для написания удобного API с готовой сериализацией и методами для настройки маршрутизации.

В качестве алгоритма генерации сначала я хотел использовать микросервис, написанный на gRPC, но в последствии отказался от этой идеи, и добавил к Django приложению брокер сообщений redis и очередь задач Celery, на которой и будет размещен алгоритм генерации. Они также написаны на Python, не придется использовать 2 разных языка и переносить модели в другой язык.

Также нам потребуется Docker, для объединения всех этих технологий в одну сеть из изолированных контейнеров, где они гарантированно запустятся, вне зависимости от операционной системы пользователя.

1. Практическая часть
   1. Техническое задание

В качестве технического задания по информационной системе для автоматического проектирования расписания были получены следующие указания:

Цель: Создание информационной системы для генерации расписания колледжа по установленным входным данным (учителя, количество предметов, кабинеты, группы)

Задачи:

— Анализ требований к информационной системе для автоматического проектирования расписания

— Установка необходимых программ и компонентов системы для проектирования информационной системы автоматического проектирования расписания

— Создание базового Django проекта для проектирования взаимодействия моделей и алгоритма генерации расписания

— Создание базы данных, используя готовые контейнеры Docker

— Настройка подключения базы данных к Django проекту

— Создание моделей объектов базы данных для информационной системы автоматического проектирования расписания

— Написание алгоритма генерации и сохранения расписания в Базе данных

Функциональные требования к информационной системе:

— Учет и хранение информации об учителях, предметах, группах

— Учет и хранение информации о готовых расписаниях

— Возможность создания нового расписания из существующих данных об учителях, предметах и группах

— Возможность задавать необходимые параметры для создания расписания (количество предметов у группы, выбор необходимых групп)

— Реализация удобного добавления данных в информационную систему с интуитивно понятным пользовательским интерфейсом и функциональными возможностями для просмотра и редактирования существующих данных

Технические требования:

— Система должна быть реализована на базе современных технологий и функциональных решениях

— Система должна быть масштабируемой, легко дополняемой к другим информационным системам

— Система должна быть отказоустойчивой и обеспечивать надежное хранение и отображение данных

— АРМ должен быть легко настраиваемым и поддерживаемым

Ожидаемый результат: В результате выполнения проекта будет создан веб сервер с удобным пользовательским интерфейсом для управления существующими данными. Также должна быть команда веб серверу, которая будет запускать алгоритм генерации расписания и сохранение его в базу данных для последующего использования и редактирования. Это позволит ускорить работу людей, которым приходится тратить время на составление расписания в ручном режиме, проверяя его правильность.

* 1. Установка программ для работы, создание Django проекта

Для успешного выполнения технического задания нам потребуется установить следующие программы:

— Python 3.10.6 <https://www.python.org/downloads/release/python-3106/> - это высокоуровневый интерпретируемый язык программирования, который отличается простотой и читаемостью кода, широким спектром применения и обширной стандартной библиотекой, предоставляющей множество готовых инструментов и модулей для различных задач разработки программного обеспечения.

* PyCharm Community <https://www.jetbrains.com/pycharm/download/> - это бесплатная интегрированная среда разработки (IDE) для языка программирования Python, предоставляющая различные функциональные возможности и инструменты для удобной разработки, отладки и управления проектами на Python.
* Docker Desktop <https://www.docker.com/products/docker-desktop/> - это инструмент, который позволяет разработчикам создавать, развертывать и управлять контейнерами на своем компьютере. Он предоставляет удобный интерфейс для работы с Docker контейнерами, позволяет легко создавать образы, запускать контейнеры, настраивать сети и хранилища, а также управлять всеми аспектами контейнеризации на локальной машине.
* DataGrip <https://www.jetbrains.com/datagrip/download/> - это интегрированная среда разработки для работы с базами данных, разработанная компанией JetBrains. Она предоставляет удобный интерфейс и мощные инструменты для подключения, обзора, редактирования и выполнения запросов к различным типам баз данных, включая SQL-базы данных, NoSQL-хранилища, облачные сервисы и другие.

После прохождения установки всех программ и перезагрузки системы, программы готовы к работе. Нам нужно создать веб сервер на Django, который будет использоваться как фреймворк для нашей информационной системы. Фреймворк - это набор предопределенных правил, структур и инструментов, предназначенных для разработки программного обеспечения. Он предоставляет основу для создания приложений, определяет архитектурные принципы, стандарты кодирования, управление зависимостями и другие соглашения, упрощающие разработку, предоставляет готовые компоненты и модули, которые можно использовать для реализации конкретных функций, что позволяет не тратить время на основные задачи, которые пришлось бы реализовывать самому.

Для начала откроем PyCharm и попадем в меню выбора проектов (Рисунок 2).

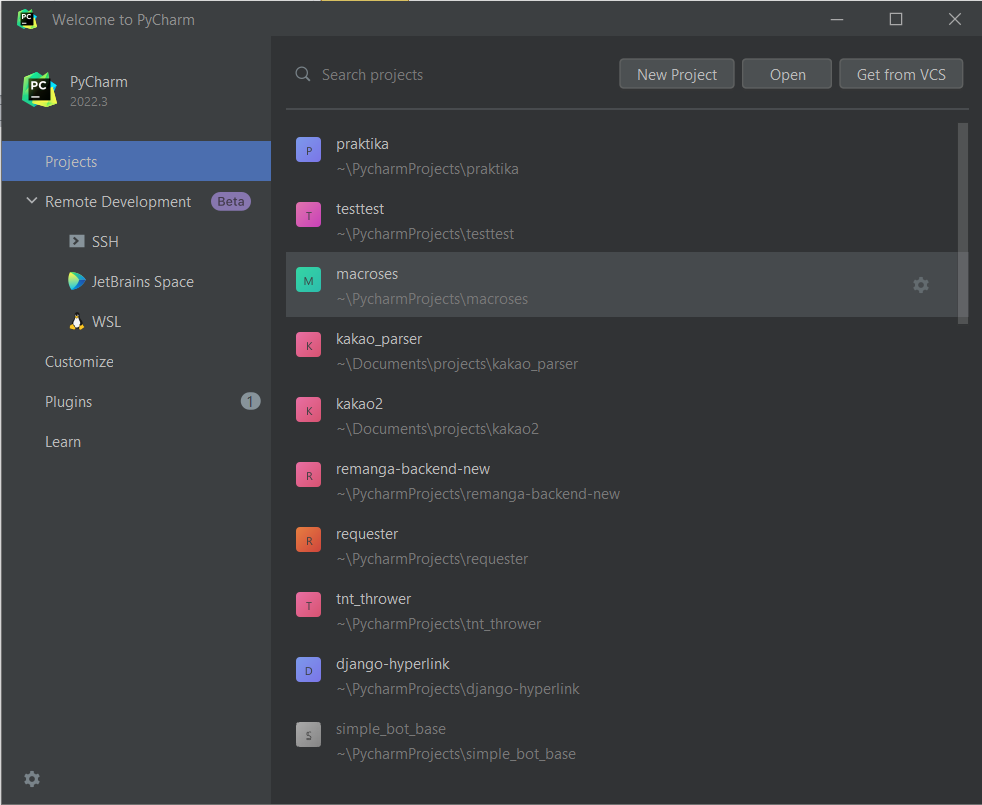


Рисунок 2 - окно выбора проектов PyCharm

Теперь нам нужно создать Django проект. Для этого кликаем по кнопке New Project в правом верхнем углу программы. В появившемся окне выбираем тип проекта Django. Сам проект я назвал «kursovaya» (Рисунок 3)

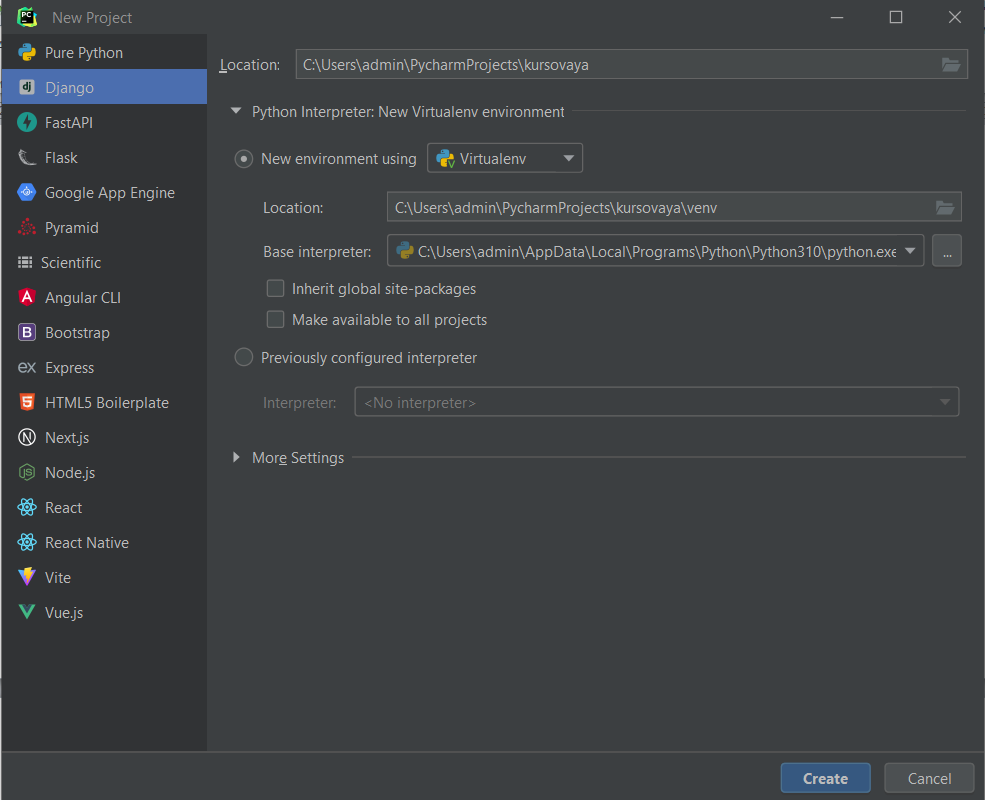


Рисунок 3 - создание Django проекта

После нажатия кнопки «Create» у меня появился новый Django проект. Сразу создадим главное приложение, в котором у нас будет основной код приложения. Назовем его «app». Выполняем команду python manage.py startapp app, где app – название приложения. Оно не может пересекаться с названиями уже установленных модулей. После выполнения структура нашего Django приложения будет выглядеть так (Рисунок 4):

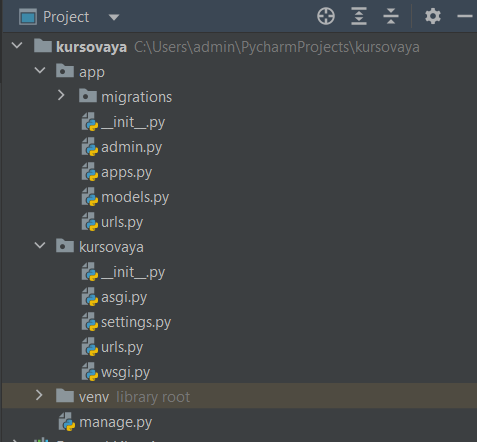


Рисунок 4 - структура приложения Django

Для того чтобы подключить наше приложение «app» к нашему Django проекту, нам нужно добавить его название в переменную INSTALLED\_APPS файла settings.py в папке с настройками проекта (Рисунок 5)

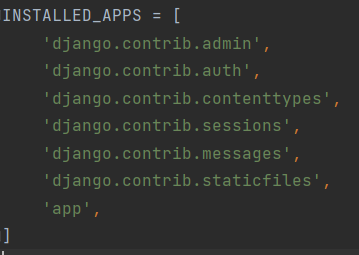


Рисунок 5 - подключенные к Django приложения

Мы успешно создали Django проект и подключили к нему приложение, в котором будет основная логика работы. Теперь нужно подключить базу данных, где будут храниться данные нашего приложения.

* 1. Создание и подключение базы данных, настройка моделей

В качестве базы данных я буду использовать СУБД PostgreSQL, так как я часто работаю с ней, и она полностью удовлетворяет все мои требования к отдельной СУБД. В качестве альтернатив можно использовать MySQL или sqlite 3. Для создания базы данных я создал docker-compose.yaml файл и написал в него контейнер для запуска базы данных, которая будет доступна локально на порту 5432 (Рисунок 6)

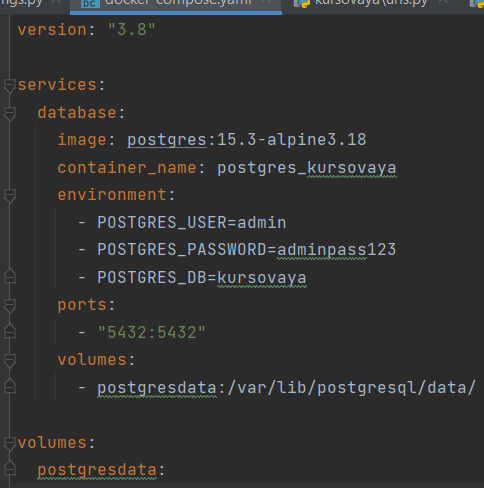


Рисунок 6 - создание базы данных

При запуске контейнера это локально создаст базу данных с пользователем «admin» и паролем «adminpass123», а также даст ему все привилегии над базой данных с названием «kursovaya». Запускам командой docker-compose up –d. После запуска проверим соединение с базой данных, для этого откроем DataGrip и создадим новое подключение к базе данных PostgreSQL (Рисунок 7)

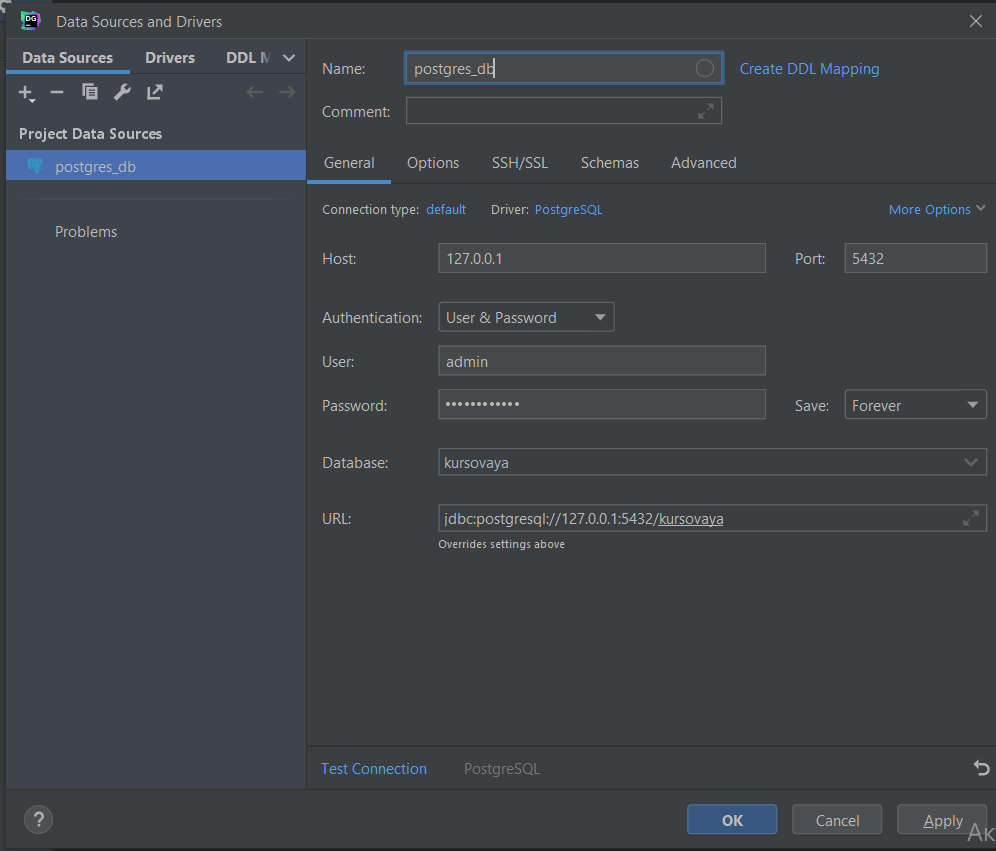


Рисунок 7 - подключение к базе данных PostgreSQL

Нажимаем Ok и смотрим результат. В списке подключений мы подключились к нашему серверу СУБД и видим базу данных kursovaya (Рисунок 8)

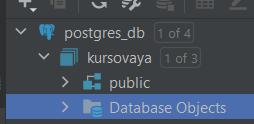


Рисунок 8 - список подключений в DataGrip

База данных готова к работе и полностью функционирует. Теперь подключим её к нашему Django проекту. Для этого в переменной DATABASES файла settings.py подключим нашу локальную базу данных (Рисунок 9)



Рисунок 9 - подключение БД к Django

Теперь нужно создать модели для каждой сущности в базе данных. Для этого опишем модели (Рисунок 10):

- Учитель (Имя, Фамилия)

- Группа (Название, список предметов)

- Предмет (Название, учитель, кабинет)

- Кабинет (Номер кабинета)

- Готовое расписание (Название, дата создания, расписание)



Рисунок 10 - Django модели сущностей

Теперь выполним команды python manage.py makemigrations и python manage.py migrate для добавления этих моделей в нашу базу данных. После выполнения обновляем подключение в DataGrip и видим, что наши модели добавлены в базу данных (Рисунок 11).

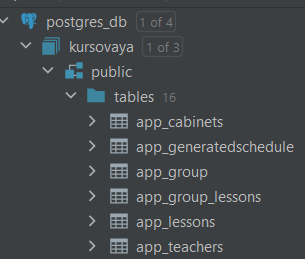


Рисунок 11 - таблицы в базе данных

Запустим веб сервер командой python manage.py runserver, перейдем по адресу <http://127.0.0.1:8000/admin/> и увидим, что мы можем создавать, изменять, просматривать и удалять наши объекты через админ панель Django. На этом настройка базы данных и создание моделей, а также создание интерфейса для их просмотра и редактирования выполнено. Теперь нужно создать алгоритм генерации, который будет создавать расписание, исходя из предоставленных данных.

* 1. Написание алгоритма генерации

Запуск алгоритма будет через запуск команды Django. Для этого, в папке нашего приложения «app» создаем пакет python management, а в нем пакет python commands. В папке commands создаем файл generate.py, а в нем создаем класс Command, унаследованный от класса BaseCommand, и функцию handle, которая и будет выполняться при запуске скрипта (Рисунок 12)

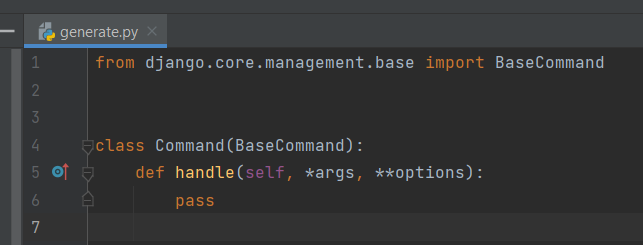


Рисунок 12 - создание Django комманды

При её запуске нам нужно получить список всех групп, спросить у пользователя, нужно ли добавлять эту группу в расписание, затем получить все предметы, у которых привязан кабинет и преподаватель, спросить сколько этих предметов будет в неделю, сохранить все введенные данные и исходя из них составить расписание. После ввода данных алгоритм проходится по всем группам, проходится по всем урока в группе, берет рандомный день, берет числа от 1 до 4, и проверяет, не стоит ли, например, в пятницу 2 предметом уже предмет, а также не стоит ли у какой-то группы в этот день и номер предмет, который занимает кабинет или учителя. Если всё окей – добавляем урок на этот день и время, и уменьшаем требуемое кол-во для генерации на 1. Если же дни и места для предмета кончились, но уроки не смогли распределиться, программа прервет работу с ошибкой, в которой будет сказано, что кабинет или учитель перегружены, и следует пересмотреть кол-во предметов в этом кабинете или у этого преподавателя. (Рисунок 13 и 14).



Рисунок 13 - ввод данных о группе и предметах

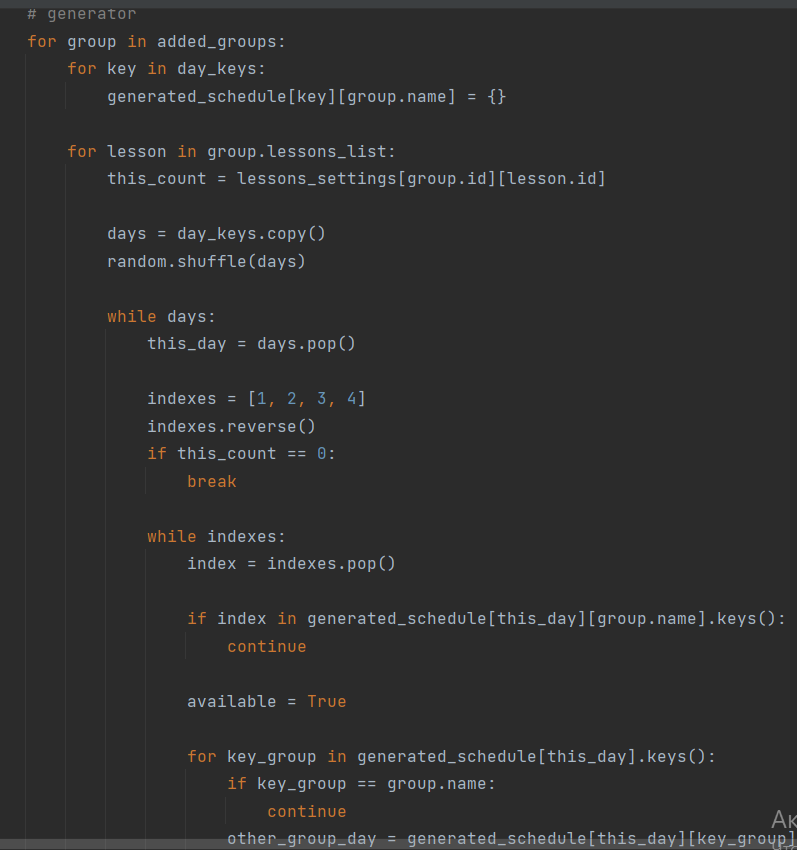


Рисунок 14 - алгоритм распределения введенных предметов

Для проверки алгоритма я заполнил базу данных случайными данными об учителях, кабинетах, преподавателях и группах. Запускаю скрипт командой python manage.py generate, ввожу данные и смотрю результат работы (Рисунок 15)

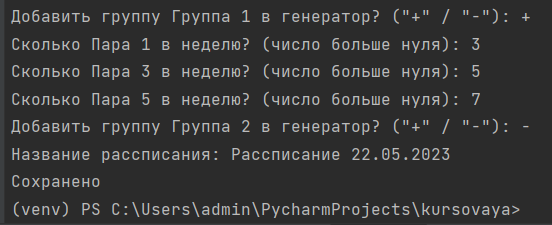


Рисунок 15 - выполнение скрипта generate

Скрипт завершился без ошибок. Теперь, чтобы проверить, сохранилось ли расписание в базе данных, нужно написать к ней запрос. Открываю DataGrip и пишу простейший запрос, для получения всех объектов из таблицы app\_generatedschedule (там, куда сохраняются расписания после генерации). Для этого я выполню SELECT запрос, где достану все поля из таблицы app\_generatedschedule (Рисунок 16)

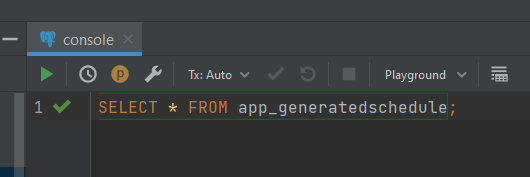


Рисунок 16 - запрос в базу данных

Выполняю запрос и смотрю результат (Рисунок 17)

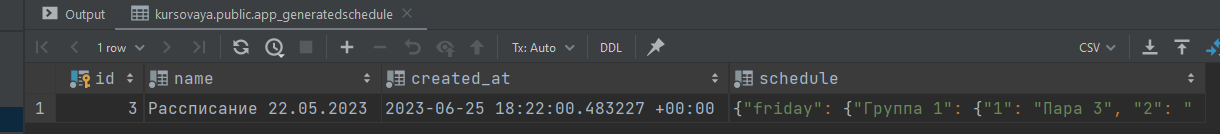


Рисунок 17 - результат запроса в базу данных

Как можно видеть, объект расписания успешно сохранен в базе данных, и доступен для просмотра, редактирования и удаления. Чтобы посмотреть сам расписание, нужно скопировать столбец schedule и посмотреть на него. Он имеет структуру:

{day: {group\_name: {lesson\_index: lesson\_name}}}

Чтобы посмотреть данные в более читаемом виде, я использовал сайт <https://codebeautify.org/jsonviewer>, куда вставил данные из столбца schedule. Сайт вывел данные в красивом и читаемом виде, где видно, в какой день у какой группы какой предмет по счету (Рисунок 18)

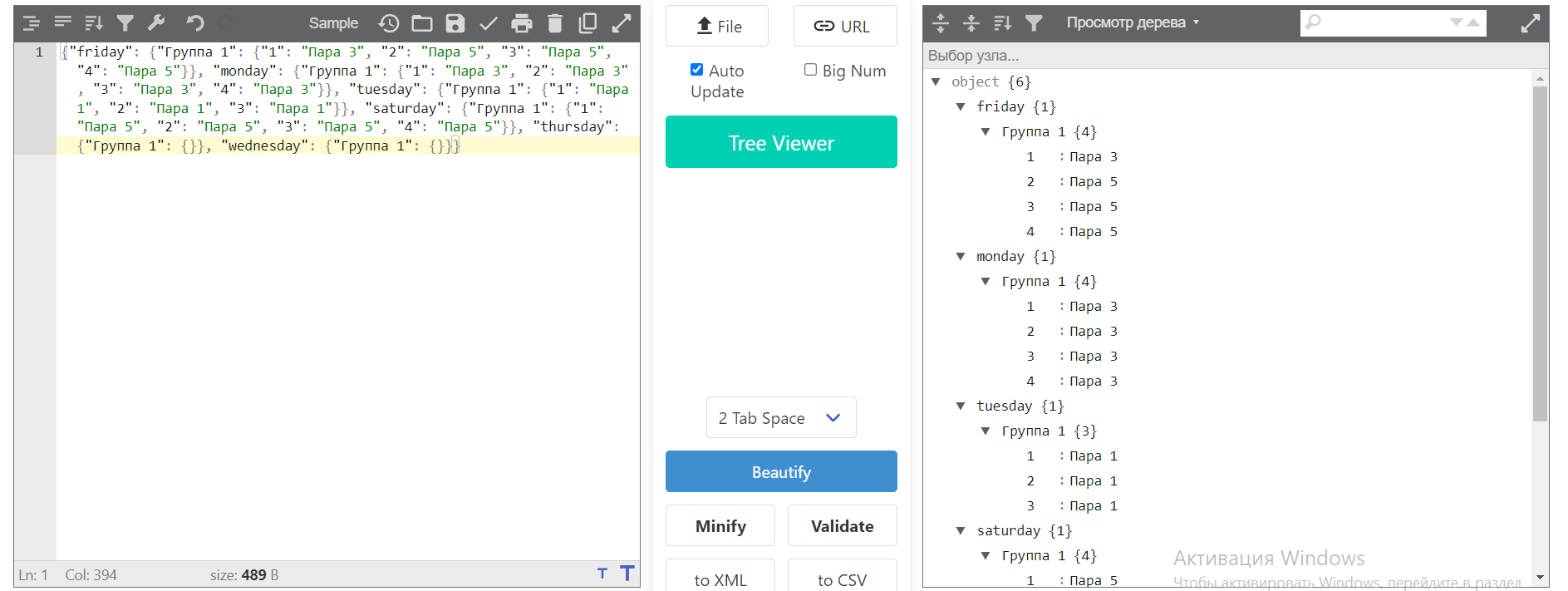


Рисунок 18 - просмотр данных о расписании

Но, допустим, я допустил ошибку в названии расписания. Для того, чтобы его обновить, мне потребуется выполнить новый SQL запрос в базу данных. На этот раз это будет UPDATE запрос. Мы обновим поле name на новое у расписания под id = 3. И также выполним SELECT запрос ещё раз, чтобы получить актуальные данные с базы данных, и проверить выполнение UPDATE запроса (Рисунок 19)

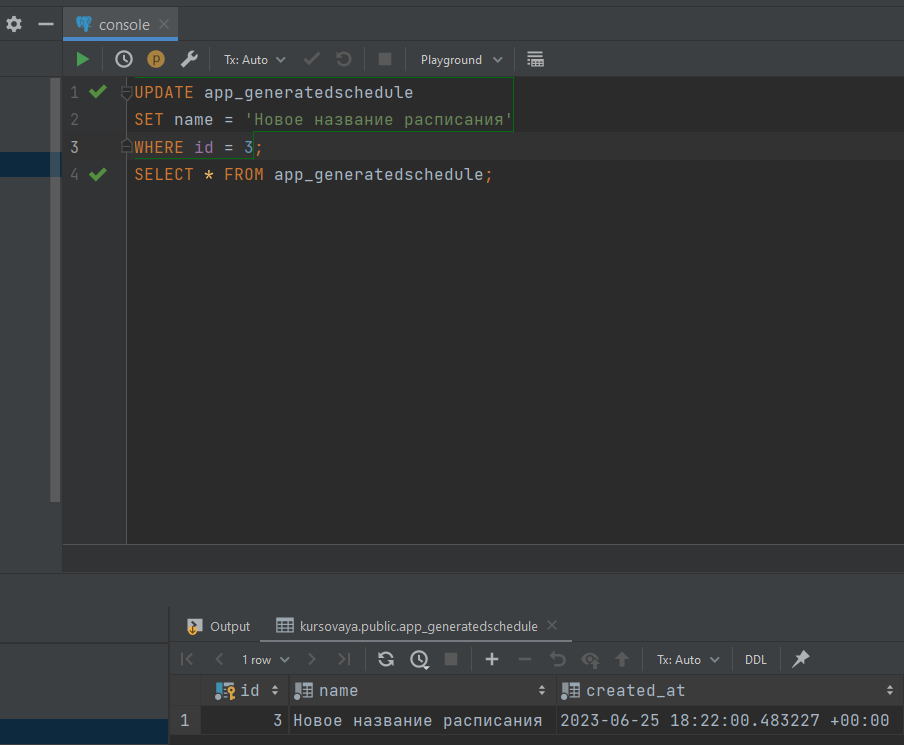


Рисунок 19 - результат выполнения нескольких запросов

Как можно видеть по галочкам слева от запросов, оба запроса успешно были выполнены, и название сменилось на «Новое название расписания». Теперь удалим это расписание, чтобы оно не путало пользователей, так как мы генерировали его только для проверки алгоритма. Для этого выполним DELETE запрос, и удалим объект с id = 3 (Рисунок 20)

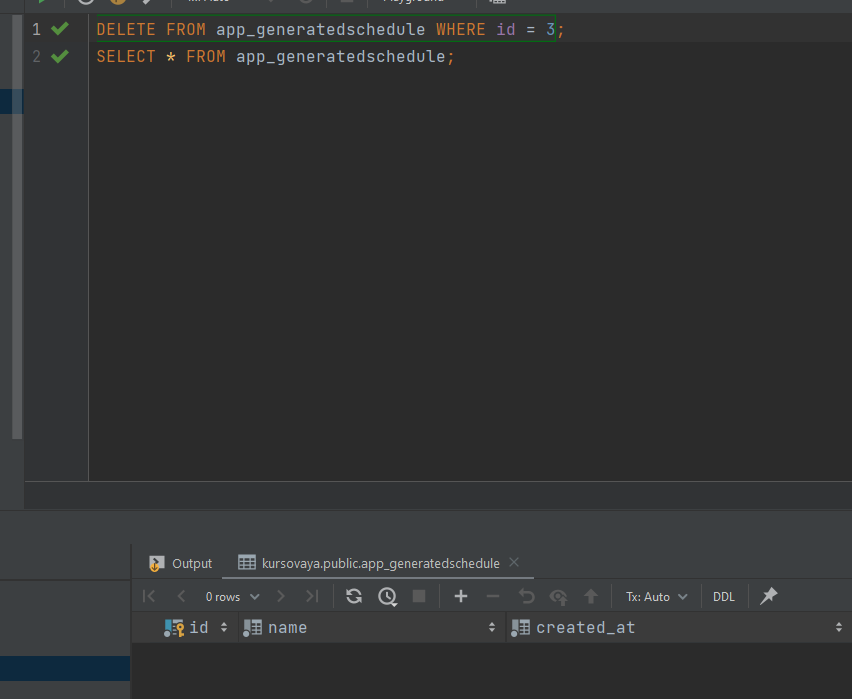


Рисунок 20 - результат удаления рассписания

Запросы выполнены, и теперь в таблице app\_generatedschedule 0 объектов.

В результате я создал веб сервер, с удобным интерфейсом управления данными, подключил надежное защищенное хранилище, создал уникальный алгоритм генерации расписания для автоматизации этого процесса и экономии времени у человека, которому приходится самому проверять правильность данных, пересечения учителей и кабинетов.

## ЗАКЛЮЧЕНИЕ

В рамках этой курсовой работы мы создали приложение, для генерации расписания по заданным параметрам и группам. Это приложение предоставляет значительные преимущества для образовательного учреждения, повышая эффективность и точность процесса составления расписания, а также облегчая работу сотрудников, занимающихся управлением учебно-методическим процессом.

Проектирование АРМ сотрудника учебно-методического отдела позволяет сотрудникам эффективно управлять и контролировать учебно-методический процесс. АРМ предоставляет доступ к централизованной базе данных, содержащей информацию о преподавателях, учебных группах, предметах и расписаниях. Это упрощает процесс составления расписания, позволяет быстро реагировать на изменения и обеспечивает своевременное информирование студентов и преподавателей о расписании занятий.

В будущем можно улучшить проект: доработать алгоритм генерации, отображение, настройки генератора, добавить ограничение определенного предмета в день (например, чтобы не ставилось 3 физкультуры в 1 день, а была одна физкультура в 3 разных дня). Также доработать скачивание расписания в формате excel, добавление названия расписания, генерация расписания на месяц вперед, привязка дней месяца к дням недели и другое.

В результате, наш проект представляет собой удобный и мощный инструмент для автоматизации процесса создания расписания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Django

<https://docs.djangoproject.com/en/4.2/>

1. Ответы пользователей со StackOverflow <https://stackoverflow.com/questions/tagged/python/>
2. Ответы пользователей Reddit

<https://reddit.com/>

1. Docker Image для PostgreSQL

<https://hub.docker.com/_/postgres/>

1. Курс молодого бойца PostgreSQL

<https://habr.com/ru/articles/340460/>

1. Установка и настройка PostgreSQL в Docker <https://timeweb.cloud/tutorials/postgresql/postgresql-docker-setup>
2. Создание первого проекта Django <https://ivanovskiy999.gitbook.io/django/1.-vvedenie-v-django/sozdanie-pervogo-proekta>
3. Книга «Python, Django и PyCharm для начинающих» Анатолий Постолит

<https://www.litres.ru/book/anatoliy-postolit/python-django-i-pycharm-dlya-nachinauschih-67727943/>

ПРИЛОЖЕНИЕ 1

КОД НА PYTHON СКРИПТА ДЛЯ ГЕНЕРАЦИИ РАССПИСАНИЯ

added\_groups = []

lessons\_settings = {}

day\_keys = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday']

generated\_schedule = {}

for key in day\_keys:

generated\_schedule[key] = {}

groups = Group.objects.prefetch\_related('lessons').all()

for group in groups:

while True:

is\_add = input(f'Добавить группу {group.name} в генератор? ("+" / "-"): ')

if is\_add in ('+', '-'):

break

print('Должно быть + или -')

if is\_add == '-':

continue

lessons = list(group.lessons.exclude(teacher\_id\_\_isnull=True, cabinet\_id\_\_isnull=True))

if len(lessons) == 0:

print(f'У группы {group.name} нет предметов, скип')

continue

setattr(group, 'lessons\_list', lessons)

added\_groups.append(group)

lessons\_settings[group.id] = {}

for lesson in lessons:

while True:

count = input(f'Сколько {lesson.name} в неделю? (число больше нуля): ')

if not count.isdigit():

print('Должно быть числом')

continue

if (count := int(count)) <= 0:

print('Должно быть больше нуля')

continue

lessons\_settings[group.id][lesson.id] = count

break

name = input('Название рассписания: ')

# generator

for group in added\_groups:

for key in day\_keys:

generated\_schedule[key][group.name] = {}

for lesson in group.lessons\_list:

this\_count = lessons\_settings[group.id][lesson.id]

days = day\_keys.copy()

random.shuffle(days)

while days:

this\_day = days.pop()

indexes = [1, 2, 3, 4]

indexes.reverse()

if this\_count == 0:

break

while indexes:

index = indexes.pop()

if index in generated\_schedule[this\_day][group.name].keys():

continue

available = True

for key\_group in generated\_schedule[this\_day].keys():

if key\_group == group.name:

continue

other\_group\_day = generated\_schedule[this\_day][key\_group]

if index not in other\_group\_day.keys():

continue

other\_group\_lesson = other\_group\_day[index]

if other\_group\_lesson.cabinet\_id == lesson.cabinet\_id or other\_group\_lesson.teacher\_id == lesson.teacher\_id:

available = False

break

if not available:

continue

generated\_schedule[this\_day][group.name][index] = lesson

this\_count -= 1

if this\_count == 0:

break

if this\_count > 0:

raise Exception(f'Невозможно сгенерировать расписание с такими настройками, \n'

f'{lesson.name}: учитель или кабинет заняты')

for day in day\_keys:

for key\_group in generated\_schedule[day]:

for index in generated\_schedule[day][key\_group].keys():

generated\_schedule[day][key\_group][index] = generated\_schedule[day][key\_group][index].name

GeneratedSchedule.objects.create(

name=name,

schedule=generated\_schedule

)

print('Сохранено')

ПРИЛОЖЕНИЕ 2

КОД DOCKER-COMPOSE ДЛЯ СОЗДАНИЯ POSTGRESQL БАЗЫ ДАННЫХ

version: "3.8"

services:

database:

image: postgres:15.3-alpine3.18

container\_name: postgres\_kursovaya

environment:

- POSTGRES\_USER=admin

- POSTGRES\_PASSWORD=adminpass123

- POSTGRES\_DB=kursovaya

ports:

- "5432:5432"

volumes:

- postgresdata:/var/lib/postgresql/data/

volumes:

postgresdata:

ПРИЛОЖЕНИЕ 3

GIT ХРАНИЛИЩЕ ВСЕХ ФАЙЛОВ КУРСОВОГО ПРОЕКТА

<https://github.com/nelttjen/kursovaya>

**ОТЗЫВ**

**РУКОВОДИТЕЛЯ КУРСОВЫМ ПРОЕКТОМ**

Студента курса , специальности

форма обучения,

На курсовой проект по теме

Описание работы

Положительные стороны

Замечания

Рекомендации

Руководитель (ФИО)

(подпись)

Дата« » 20 г.