

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего  
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, системы, комплексы и  
сети»

БАКАЛАВРСКАЯ РАБОТА

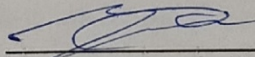
на тему

**WEB-ПРИЛОЖЕНИЕ СОСТАВЛЕНИЯ**  
**РАСПИСАНИЯ ВУЗА**

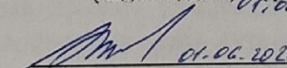
Студент

 01.06.2026 Курушин Ярослав Сергеевич  
(подпись, дата) (ФИО полностью)

Руководитель

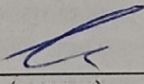
 Зинкин С.А.  
(подпись, дата) (фамилия, инициалы)

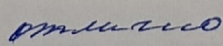
Нормоконтролёр

 01.06.2026 Бычков А.С.  
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

 Митрохин М.А.  
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от 15.06.2026 № 13)

Секретарь ГЭК

 Гурин Е.И.  
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

«Утверждаю»  
Заведующий кафедрой  
М.А. Митрохин  
«7» апреля 2026 г.

**ЗАДАНИЕ  
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Курушин Ярослав Сергеевич гр.22BVB1 факультета вычислительной техники

2. Тема работы: «Web-приложение составления расписания ВУЗа»

Тема утверждена приказом ПГУ № 347/о от «7» апреля 2026 г.

3. Руководитель бакалаврской работы д.т.н., профессор кафедры ВТ ПГУ Зинкин С.А.

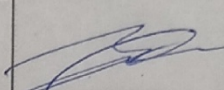
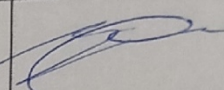
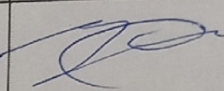
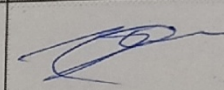
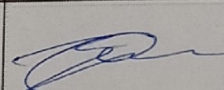
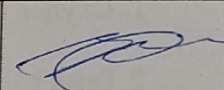
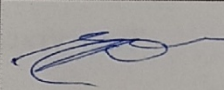
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать «Web-приложение составления расписания ВУЗа». Приложение должно обладать графическим и интуитивно понятным интерфейсом для взаимодействия с пользователем. Система должна предоставлять возможность для работы с информацией о пользователях, ролях, и поддерживать аутентификацию. Разрабатываемое приложение должно выполнять задачи по загрузке и парсингу поручений, создание, редактирование, удаление занятий.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области систем составления расписаний: изучить существующие решения, выделить их достоинства и недостатки. Выбрать язык программирования, серверный фреймворк и технологию для создания графического интерфейса. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для работы с файловой системой, для реализации аутентификации и для взаимодействия с базой данных.

6. Календарный график выполнения работы

| № п/п | Наименование этапов работы                                  | Объем работы | Срок выполнения         | Подпись руководителя                                                                  |
|-------|-------------------------------------------------------------|--------------|-------------------------|---------------------------------------------------------------------------------------|
| 1     | Изучение предметной области                                 | 30           | 07.04.2026 – 10.04.2026 |    |
| 2     | Анализ требований к платформе                               | 24           | 10.04.2026 – 13.04.2026 |    |
| 3     | Выбор технологий, инструментов и проектирование архитектуры | 8            | 13.04.2026 – 14.04.2026 |    |
| 4     | Создание макета графического интерфейса                     | 8            | 14.04.2026 – 20.04.2026 |    |
| 5     | Разработка серверной логики                                 | 60           | 20.04.2026 – 07.05.2026 |    |
| 6     | Реализация клиентского интерфейса                           | 55           | 07.05.2026 – 14.05.2026 |  |
| 7     | Оформление пояснительной записки                            | 20           | 14.05.2026 – 21.05.2026 |  |

Дата выдачи задания «07» апреля 2026г.

Руководитель бакалаврской работы \_\_\_\_\_ Зинкин С.А.  
(подпись, дата)

Задание к исполнению принял студент \_\_\_\_\_ Курушин Я.С.  
(подпись, дата)

Работу к защите допустить  
Декан факультета \_\_\_\_\_ Фионова Л.Р.  
(подпись, дата) 10.06.2026

## Реферат

Бакалаврская работа содержит 187 листов, 39 рисунков 1 таблицу 7 источников, 3 приложения.

WEB-ПРИЛОЖЕНИЕ, СОСТАВЛЕНИЕ РАСПИСАНИЯ, FASTAPI, REACT, POSTGRESQL, ПАРСИНГ DOCX, АУТЕНТИФИКАЦИЯ JWT, РОЛЕВАЯ МОДЕЛЬ, СОВМЕСТИМЫЕ ЗАНЯТИЯ, ЭКСПОРТ EXCEL

Цель работы – разработка Web-приложения для автоматизированного составления семестрового расписания учебных занятий вуза на основе файла поручений в формате Microsoft Word.

Для достижения поставленной цели спроектирована и реализована система, обеспечивающая импорт структурированных данных из Word-документов, ролевое разграничение доступа (администратор и преподаватель), интерактивное создание и редактирование расписания с автоматической валидацией конфликтов (занятость аудиторий, групп, преподавателей, дневной лимит пар, соответствие оборудования). Реализован механизм совместных лабораторных занятий с поддержкой приглашений и разделения групп на подгруппы. Разработаны модули экспорта в Excel (общая преподавательская ведомость и студенческое расписание по группам). Проведено тестирование, подтвердившее корректность работы всех функций системы.

Внедрение разработанного приложения позволяет снизить временные затраты на формирование и согласование расписания, исключить ошибки, связанные с человеческим фактором, и повысить вовлечённость преподавателей в процесс планирования учебной нагрузки.

|                                                                     |      |               |                                                                                     |          |                            |      |        |
|---------------------------------------------------------------------|------|---------------|-------------------------------------------------------------------------------------|----------|----------------------------|------|--------|
| ПГУ 1.090301.06.001 ПЗ                                              |      |               |                                                                                     |          |                            |      |        |
| Изм.                                                                | Лист | № докум       | Подпись                                                                             | Дата     | Литера                     | Лист | Листов |
| Разраб                                                              |      | Курушин Я.С.  |  | 01.06.20 |                            | 2    | 187    |
| Пров                                                                |      | Зинкин С.А.   |  | 01.06.20 |                            |      |        |
| Н. Контр.                                                           |      | Бычков А.С.   |  | 01.06.20 | ФВТ, каф. ВТ<br>гр. 22ВВВ1 |      |        |
| Утв                                                                 |      | Митрохин М.А. |  | 01.06.20 |                            |      |        |
| Web-приложение составления расписания ВУЗа<br>Пояснительная записка |      |               |                                                                                     |          |                            |      |        |

## Содержание

|                                                             |     |
|-------------------------------------------------------------|-----|
| Перечень сокращений и обозначений.....                      | 7   |
| Введение.....                                               | 9   |
| 1 Web-приложение составления расписания ВУЗа.....           | 11  |
| 2 Анализ предметной области.....                            | 11  |
| 1.2 Анализ актуальности разработки.....                     | 13  |
| 1.3 Анализ аналогов информационных систем.....              | 15  |
| 1.4 Постановка задачи.....                                  | 22  |
| 2 Выбор технологий и средств разработки приложения.....     | 26  |
| 3 Реализация.....                                           | 38  |
| 3.1 Архитектура приложения.....                             | 38  |
| 3.2 Работа с базой данных.....                              | 40  |
| 3.3 Разработка клиентского приложения.....                  | 43  |
| 3.3.1 Компонент сетки расписания.....                       | 44  |
| 3.3.2 Модальное окно SlotModal.....                         | 44  |
| 3.3.3 Аутентификация и маршрутизация.....                   | 45  |
| 3.4 Разработка серверного приложения.....                   | 50  |
| 3.4.1 Модуль аутентификации и авторизации.....              | 50  |
| 3.4.2 Модуль импорта поручений (docx_parser).....           | 50  |
| 3.4.3 Модуль валидации расписания (schedule_validator)..... | 51  |
| 3.4.4 Роутеры.....                                          | 53  |
| 3.4.5 Экспорт в Excel.....                                  | 55  |
| 3.4.6 Административные функции.....                         | 56  |
| 3.5 Тестирование.....                                       | 56  |
| 4 Охрана труда.....                                         | 82  |
| 4.1 Организация рабочего места.....                         | 82  |
| 4.2 Режим труда и профилактика утомления.....               | 83  |
| 4.3 Электробезопасность и пожарная безопасность.....        | 83  |
| Заключение.....                                             | 85  |
| Список использованных источников.....                       | 88  |
| Приложение А.....                                           | 89  |
| Приложение Б.....                                           | 98  |
| Приложение В.....                                           | 251 |

## **Перечень сокращений и обозначений**

В настоящей работе применяют следующие обозначения и сокращения

БД - база данных

ОС - операционная система

ПК - персональный компьютер

СУБД - система управления базами данных

API (Application Programming Interface) - набор правил и интерфейсов, позволяющий различным программным компонентам обмениваться данными и взаимодействовать друг с другом

URL (Uniform Resource Locator) - это уникальный адрес, который указывает на расположение ресурса (сайта, файла, изображения) в интернете

CSS (Cascading Style Sheets) - язык описания внешнего вида веб-страниц, отвечающий за оформление и расположение элементов

HTML (HyperText Markup Language) - язык разметки, применяемый для структурирования содержимого веб-страниц

IDE (Integrated development environment) - интегрированная среда разработки

JS (JavaScript) - язык программирования, используемый преимущественно для создания интерактивных элементов веб-страниц

VS Code (Visual Studio Code) - бесплатный редактор исходного кода от компании Microsoft, созданный для Windows, macOS и Linux

SQL (Structured Query Language) - язык структурированных запросов

UML (Unified Modeling Language) - унифицированный язык моделирования

CSV (Comma-Separated Values) - это текстовый формат для представления табличных данных

PDF (Portable Document Format) - это популярный универсальный формат электронных документов

HTTP (HyperText Transfer Protocol) - это базовый протокол передачи данных в интернете

CRUD (Create, Read, Update, Delete) - это аббревиатура из четырех базовых операций для работы с данными.

## Введение

Составление семестрового расписания учебных занятий в высшем учебном заведении традиционно считается одной из наиболее трудоёмких административных задач. Её сложность обусловлена необходимостью одновременного учёта множества разнородных факторов: распределённой по кафедрам и корпусам учебной нагрузки, ограниченного аудиторного фонда с различным уровнем материально-технического оснащения, индивидуальных методических дней и пожеланий сотен преподавателей, а также санитарно-гигиенических норм, регламентирующих предельную дневную нагрузку студентов. На практике, несмотря на повсеместную цифровизацию, многие вузы продолжают использовать для решения этой задачи связку из электронных таблиц и ручной проверки, что неизбежно приводит к возникновению ошибок, многократным итерациям согласования и значительным временным затратам диспетчеров учебных отделов.

Анализ существующих на рынке программных продуктов показал, что крупные корпоративные системы (1С:Университет, Галактика ВУЗ) требуют существенных финансовых вложений и сложны в адаптации под специфический документооборот конкретного вуза, в то время как специализированные приложения («Экспресс-расписание», aSc TimeTables) ориентированы преимущественно на школы и небольшие колледжи, не поддерживают полноценную многопользовательскую работу и не имеют средств автоматического импорта нагрузки из произвольных документов формата Microsoft Word. Ключевым недостатком всех рассмотренных аналогов является централизованная модель составления расписания, при которой вся полнота ответственности и трудозатрат ложится на диспетчеров, а преподаватели выступают лишь в роли пассивных наблюдателей, вынужденных затем инициировать многочисленные переносы.

Целью дипломного проектирования является создание программного продукта, позволяющего автоматизировать импорт данных из файлов поручений,

обеспечить преподавателям возможность самостоятельного выбора времени и места проведения занятий в рамках заданных ограничений, а администраторам - централизованный контроль и утверждение итогового документа.

## **1 Web-приложение составления расписания ВУЗа**

### **1.1 Анализ предметной области**

Предметной областью разработки является деятельность по планированию и оперативному управлению учебным процессом на уровне высшего учебного заведения. В частности, рассматривается процесс формирования семестрового расписания учебных занятий на основании документа, именуемого «Поручения преподавателям кафедры». Данный документ является основным нормативно-распорядительным актом, регламентирующим закрепление учебных дисциплин, видов учебной работы и контингента студентов за конкретными преподавателями в рамках утверждённого учебного плана и штатного расписания профессорско-преподавательского состава (ППС).

Контингент обучающихся включает несколько десятков академических групп различных курсов и форм обучения (очная, очно-заочная, заочная). Номенклатура дисциплин охватывает широкий спектр тем: от фундаментальных основ вычислительной техники («Арифметические и логические основы ВТ», «Теория автоматов», «Электроника и схемотехника») до современных прикладных технологий («Технологии глубоких нейронных сетей», «Разработка кроссплатформенных приложений», «Web-программирование»).

Процесс составления расписания на основе поручений включает решение задачи удовлетворения ограничений. Основные ограничения можно классифицировать следующим образом:

Пространственные ограничения: одна аудитория в один момент времени может использоваться только для одного занятия. Для лабораторных работ необходимо, чтобы в аудитории было установлено требуемое количество персональных компьютеров с необходимым программным обеспечением. Для лекций в большинстве случаев требуется наличие мультимедийного проектора и экрана (или интерактивной доски).

Временные ограничения: для каждой академической группы дневная аудиторная нагрузка не должна превышать четырёх пар (восьми академических

часов), что соответствует санитарно-гигиеническим нормам. Недопустимо наличие «окон» в расписании группы, хотя это ограничение не всегда является жёстким.

Субъектные ограничения: один преподаватель не может одновременно вести два разных занятия. Кроме того, существуют персональные ограничения, указанные в примечаниях к поручениям: для ряда преподавателей недопустимы занятия в субботу, в ранние утренние часы, в определённые дни недели. Для других преподавателей, напротив, рекомендовано планировать занятия в определённые дни. Эти индивидуальные пожелания и методические дни значительно усложняют задачу ручного составления расписания и требуют особого учёта при автоматизации.

Существующий процесс составления расписания носит преимущественно ручной или полуавтоматизированный характер. Диспетчер (отдел управления ПГУ) получает файл поручений в формате Microsoft Word и вручную, с использованием электронных таблиц, формирует сетку расписания на семестр. При этом проверка вышеперечисленных ограничений выполняется визуально или с помощью простейших формул, что не гарантирует отсутствия коллизий. Особенно остро проблема проявляется на этапе согласования расписания с преподавателями, когда выясняется, что некоторые занятия поставлены в неприемлемое для них время, и требуется многократное ручное перекраивание сетки.

Таким образом, предметная область характеризуется высокой структурной сложностью, наличием большого числа разнородных сущностей и многоуровневых ограничений, что делает её благодатной почвой для внедрения автоматизированной информационной системы поддержки принятия решений.

## **1.2 Анализ актуальности разработки**

Разработка Web-приложения для автоматизированного составления расписания ВУЗа является актуальной и практически значимой задачей, что подтверждается рядом факторов как общего характера, так и специфических для данной предметной области.

Снижение трудоёмкости и временных затрат административно-управленческого персонала. В условиях современного университета с большим количеством направлений подготовки, профилей и постоянно обновляющимися учебными планами задача составления непротиворечивого расписания становится всё более ресурсоёмкой. На составление расписания одной кафедры на семестр может уходить от 2 до 4 недель рабочего времени диспетчера. Автоматизация рутинных проверок, таких как поиск свободной аудитории с нужным оборудованием или контроль занятости преподавателя, позволяет сократить эти затраты на 60–70%, высвобождая время для решения более творческих и стратегических задач.

Минимизация ошибок, обусловленных человеческим фактором. Ручное составление расписания неизбежно сопряжено с риском возникновения накладок. Наиболее распространённые ошибки включают: назначение двух разных занятий в одну и ту же аудиторию одновременно; направление группы в аудиторию, не оснащённую необходимым оборудованием.

Автоматизированная система, выполняющая жёсткую валидацию каждого вносимого изменения, полностью исключает возможность сохранения противоречивых данных, гарантируя корректность итогового расписания.

Децентрализация процесса и повышение вовлечённости преподавателей. Традиционная модель предполагает, что диспетчер единолично формирует расписание, а затем рассылает его преподавателям для ознакомления и внесения корректировок. Этот процесс порождает длительную переписку и множество конфликтных ситуаций. Предлагаемая система меняет парадигму: преподаватель получает инструмент для самостоятельного планирования своей нагрузки в рамках заданных ограничений (доступные группы, часы, разрешённые аудитории). Это не только снижает нагрузку на диспетчера, но и повышает удовлетворённость преподавателей, так как они получают возможность влиять на своё расписание. Роль администратора сводится к контролю и утверждению итогового документа, что соответствует современным принципам децентрализованного управления.

Учёт специфики документооборота. Универсальные программные продукты для составления расписания (например, «1С:Университет», «Галактика ВУЗ») слишком комплексные. В ВУЗе сложилась уникальная практика оформления поручений: использование сложной табличной вёрстки в Microsoft Word, включение в ячейки подробных комментариев о наличии проекторов в конкретных аудиториях, указание персональных ограничений преподавателей в произвольной текстовой форме. Интеграция с такими крупными системами потребовала бы существенной доработки или ручного ввода данных, что нивелирует преимущества автоматизации. Разработка специализированного приложения, «заточенного» под данный формат поручений, позволяет автоматизировать именно те процессы, которые наиболее проблемны.

Разработка Web-приложения для составления расписания ВУЗа является своевременной, востребованной и практически значимой задачей, решение которой позволит повысить эффективность управления учебным процессом.

### 1.3 Анализ аналогов информационных систем

Для обоснования необходимости разработки нового программного продукта проведём сравнительный анализ существующих решений, которые могут быть использованы для автоматизации составления расписания в высшем учебном заведении. Рассмотрим аналоги.

| День        | Урок        | Помещение            | Расписание                                         |
|-------------|-------------|----------------------|----------------------------------------------------|
| Пн 14.11.16 | 08:00-09:30 | 402a                 | 42016 3D моделирование Иванов И.И.                 |
|             | 09:50-11:20 | 905a                 | 905a Бухгалтерский учет Иванов И.И.                |
|             | 11:40-13:10 | МСФО А/Ф 3 АЦ Иванов | 4200a (МСФО) МСФО Иванов И.И.                      |
|             | 14:00-15:30 | 901a                 | 901a Физическая культура (плавание) Виртуальный Ф. |
| Вт 15.11.16 | 08:00-09:30 | 402a                 | 42016 3D моделирование Иванов И.И.                 |
|             | 09:50-11:20 | 905a                 | 905a Бухгалтерский учет Иванов И.И.                |
|             | 11:40-13:10 | МСФО А/Ф 3 АЦ Иванов | 4200a (МСФО) МСФО Иванов И.И.                      |
|             | 14:00-15:30 | 901a                 | 901a Физическая культура (плавание) Виртуальный Ф. |
| Ср 16.11.16 | 08:00-09:30 | 402a                 | 42016 3D моделирование Иванов И.И.                 |
|             | 09:50-11:20 | 905a                 | 905a Бухгалтерский учет Иванов И.И.                |
|             | 11:40-13:10 | МСФО А/Ф 3 АЦ Иванов | 4200a (МСФО) МСФО Иванов И.И.                      |
|             | 14:00-15:30 | 901a                 | 901a Физическая культура (плавание) Виртуальный Ф. |
| Чт 17.11.16 | 08:00-09:30 | 402a                 | 42016 3D моделирование Иванов И.И.                 |
|             | 09:50-11:20 | 905a                 | 905a Бухгалтерский учет Иванов И.И.                |
|             | 11:40-13:10 | МСФО А/Ф 3 АЦ Иванов | 4200a (МСФО) МСФО Иванов И.И.                      |
|             | 14:00-15:30 | 901a                 | 901a Физическая культура (плавание) Виртуальный Ф. |
| Пт 18.11.16 | 08:00-09:30 | 402a                 | 42016 3D моделирование Иванов И.И.                 |
|             | 09:50-11:20 | 905a                 | 905a Бухгалтерский учет Иванов И.И.                |
|             | 11:40-13:10 | МСФО А/Ф 3 АЦ Иванов | 4200a (МСФО) МСФО Иванов И.И.                      |
|             | 14:00-15:30 | 901a                 | 901a Физическая культура (плавание) Виртуальный Ф. |

Рисунок 1 – Обзор аналога «1С:Автоматизированное составление расписаний. Университет»

Первый аналог - «1С:Автоматизированное составление расписаний. Университет».

Данный продукт представляет собой специализированную конфигурацию, разработанную на технологической платформе «1С:Предприятие 8.3» и предназначенную для комплексной автоматизации процессов планирования и составления расписания учебных занятий в организациях высшего и среднего профессионального образования.

Основные функциональные возможности и особенности: управление нормативно-справочной информацией; импорт данных об учебных планах и распределённой нагрузке из внешних систем (как правило, из «1С:Университет») или ручной ввод; формирование пула учебных поручений, содержащих связи «Дисциплина – Группа (поток) – Преподаватель – Количество часов по видам занятий»; в системе реализован эвристический алгоритм, основанный на методе последовательного назначения с элементами локальной оптимизации. Алгоритм учитывает десятки типов ограничений, которые пользователь может гибко настраивать (приоритеты, жёсткие и мягкие ограничения).

Достоинства: высокая степень интеграции с другими подсистемами вуза; поддержка многопользовательского режима и разграничения прав доступа; наличие отработанных алгоритмов автоматической генерации расписания.

Недостатки: избыточная функциональность и высокая стоимость - внедрение и сопровождение требует значительных финансовых и временных затрат; закрытость и сложность адаптации - несение изменений в алгоритмы требует привлечения квалифицированного 1С-программиста; централизованная модель ввода данных - функционал самообслуживания для преподавателей либо отсутствует, либо сильно ограничен, что не решает проблему децентрализации процесса согласования.

Вывод по продукту «1С:Автоматизированное составление расписаний. Университет»: продукт является мощным инструментом для автоматизации работы учебного управления, но является избыточным и экономически неэффективным для решения задач ВУЗа. Ключевым препятствием является

отсутствие механизма прямого импорта из специфических Word-поручений и невозможность вовлечения преподавателей в процесс формирования расписания без участия диспетчера

Второй аналог - «Экспресс-расписание ВУЗ».

Экспресс-расписание ВУЗ - [Планшет занятий 2]

Файл Правка Вид Действия Операции Справочники Отчеты Сервис Окно Справка

1 Пн 15 Январь 2018 1 Вт 16 Январь 2018 1 Ср 17 Январь 2018 Рекомендуется в день: 5,0 час. 018 1 Пт 19 Январь 2018 1 Сб 20 Январь 2018

| Преподаватель   | Группа | П/г | Дисциплина        | План. ч. | Факт. ч. | Всего. ч. | Остаток | (1)  | 2    | 3    | 4 | 5 | 6 | 7 | 8 | Аудитор.  | Свобод. | Окончание  | Тип занятия |
|-----------------|--------|-----|-------------------|----------|----------|-----------|---------|------|------|------|---|---|---|---|---|-----------|---------|------------|-------------|
| Сидорова В.С.   | Б 21   | 0   | Экономика орг.    | 5        | 2        | 74        | 72      |      |      |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Горева О.В.     | Б 21   | 0   | Теория бух. учета | 5        | 4        | 78        | 74      |      |      |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Хохлов А.Ю.     | Б 21   | 0   | Физкультура       | 2        | 2        | 30        | 28      |      |      |      |   |   |   |   |   | Спорт.зал | 1       | 30.06.2018 | Лекция      |
| Горева О.В.     | Б 21   | 0   | Бухгалтерский...  | 3        | 4        | 50        | 46      |      |      |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Шабалина Е.В.   | Б 21   | 0   | Орг. и технол...  | 2        | 0        | 40        | 40      | М 21 | М 22 |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Даровский Е.В.  | Б 21   | 0   | Онц. философия    | 3        | 2        | 42        | 40      | Б 24 | Б 23 | Б 12 |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Даровский Е.В.  | Б 21   | 0   | Основы права      | 2        | 2        | 32        | 30      | Б 24 | Б 23 | Б 12 |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Корогавова А.Н. | Б 21   | 2   | Иностранный...    | 4        | 4        | 62        | 58      | М 11 | Б 11 |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Практика    |
| Коренько Л.Д.   | Б 21   | 0   | Безопас. жизнь    | 2        | 2        | 38        | 36      | Б 23 | Б 24 |      |   |   |   |   |   | 101       | 6       | 30.06.2018 | Лекция      |
| Торикова Л.А.   | Б 21   | 1   | Иностранный...    | 4        | 4        | 62        | 58      | М 11 | Б 11 | Б 34 |   |   |   |   |   | 215       | 1       | 30.06.2018 | Практика    |
| Шабалина Е.В.   | Б 21   | 0   | Эко. осн. предпр. | 2        | 2        | 32        | 30      | М 21 | М 22 |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Кислицына Н.С.  | Б 21   | 0   | Эко. теория       | 3        | 4        | 48        | 44      | Б 22 | Б 21 |      |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |
| Герасимов С.Н.  | Б 21   | 0   | Математика        | 2        | 4        | 38        | 34      | Б 21 | Б 22 | Б 24 |   |   |   |   |   | Общего..  | 6       | 30.06.2018 | Лекция      |

Выделено ячеек: 1 10.01.2018 Неделя: 2

Рисунок 2 – Обзор аналога «Экспресс-расписание ВУЗ»

«Экспресс-расписание ВУЗ» – это самостоятельный программный продукт, разработанный российской компанией ООО «Лаборатория ММИС» и широко применяемый в средних специальных и высших учебных заведениях для автоматизации составления расписания. Программа существует в нескольких редакциях, ориентированных на разные типы учебных заведений (школа, колледж, вуз). В отличие от модуля в составе 1С, это отдельное Windows-приложение.

Основные функциональные возможности и особенности: быстрое развёртывание и локальная архитектура; разработчики заявляют об использовании метода «логического моделирования», который, в отличие от классических эвристик, имитирует процесс рассуждений опытного диспетчера.



ASC Timetables - это специализированное программное обеспечение для автоматизированного составления расписания, разработанное компанией Applied Software Consultants. Продукт существует на рынке более двух десятилетий и изначально позиционировался как решение для средних общеобразовательных школ, гимназий и лицеев, однако в последние годы разработчики расширили его применимость на колледжи и небольшие университеты.

Достоинства:

- интуитивно понятный интерфейс, не требующий специального обучения.
- быстрая автоматическая генерация с учётом множества ограничений.
- качественная визуализация и удобные средства ручной корректировки.
- наличие онлайн-версии с историей изменений и возможностью совместной работы.
- доступная ценовая политика по сравнению с корпоративными ERP-системами.

Недостатки:

- ориентация преимущественно на школьный и колледж-уровень, недостаточная масштабируемость для крупного университета.
- отсутствие полноценной поддержки децентрализованного составления расписания силами преподавателей.
- невозможность прямого импорта из Word-файлов поручений.
- ограниченная поддержка операционных систем, отличных от Windows.

Вывод по продукту «ASC Timetables»: Программа представляет собой качественное и проверенное временем решение для автоматизации составления расписания в средних учебных заведениях и небольших колледжах. Однако её архитектурные ограничения - однопользовательская модель работы, недостаточная масштабируемость для уровня крупного вуза, отсутствие средств импорта из специфических Word-поручений и невозможность организации самообслуживания преподавателей - делают её малоприспособленной для решения задачи децентрализованного составления общеузовского расписания.

Вывод по результатам анализа аналогов.

Существующие на рынке программные продукты либо являются избыточными и дорогостоящими, либо требуют значительных трудозатрат на адаптацию и ввод данных, либо не обеспечивают необходимого уровня автоматизации проверок. Ни одно из рассмотренных решений не удовлетворяет поставленным задачам (децентрализация, дешевизна, удобство). Это подтверждает целесообразность разработки специализированного Web-приложения, ориентированного на конкретные нужды ВУЗа.

#### **1.4 Постановка задачи**

Целью проекта является разработка Web-приложения для автоматизации процесса составления расписания учебных занятий ВУЗа на основе файла поручений.

Формализация требований.

Изучить структуру документа «Поручения преподавателям кафедры» и выявить закономерности размещения информации о дисциплинах, группах, преподавателях, часах и аудиториях.

Сформулировать полный перечень ограничений, которые должны соблюдаться при составлении расписания (пространственные, временные, субъектные).

Определить роли пользователей системы и сценарии их работы.

Проектирование архитектуры системы.

Разработать логическую и физическую модели реляционной базы данных, обеспечивающие хранение всех необходимых сущностей и связей.

Спроектировать REST API для взаимодействия клиентской и серверной частей, включая аутентификацию, управление данными и работы с расписанием.

Разработка модуля импорта поручений.

Реализовать на языке Python с использованием библиотеки `python-docx` парсер, извлекающий из Word-файла структурированные данные о дисциплинах, группах, преподавателях, объёме часов и предпочтительных аудиториях.

Обеспечить корректную обработку объединённых ячеек, многострочных записей и диапазонов групп (например, «25BBB1-4»).

Предусмотреть механизм сохранения извлечённых данных в базу данных с поддержкой обновления существующих записей.

Разработка серверной части.

Реализовать API для регистрации и аутентификации пользователей с использованием JWT (Access и Refresh токенов) и разграничением ролей «Администратор» и «Преподаватель».

Разработать модуль валидации элементов расписания, выполняющий в рамках одной транзакции проверку на:

- занятость аудитории;
- занятость академической группы;
- занятость преподавателя;
- превышение дневного лимита в 4 пары;
- соответствие оборудования аудитории типу занятия;
- соблюдение персональных ограничений преподавателя (дни недели, время).

Реализовать CRUD-операции для сущностей: аудитории, группы, дисциплины, назначения преподавателей, элементы расписания.

Обеспечить управление жизненным циклом расписания со сменой статусов («черновик» → «на утверждении» → «утверждён»).

Разработка клиентской части.

Создать одностраничное приложение (SPA) на React с использованием функциональных компонентов и хуков.

Реализовать интерфейс администратора:

- страницу загрузки файла поручений и просмотра результатов импорта;
- страницу управления пользователями;
- страницу просмотра и утверждения расписания преподавателей.
- Реализовать интерфейс преподавателя:
  - интерактивную матрицу расписания с визуализацией занятых и доступных слотов;
  - форму выбора дисциплины, группы и аудитории при добавлении занятия;

- механизм отправки расписания на утверждение.

Обеспечить взаимодействие с сервером через HTTP-запросы с использованием библиотеки Axios, обработку ошибок и отображение уведомлений.

На рисунке 4 приведена диаграмма вариантов использования



Рисунок 4 – Диаграмма вариантов использования

Нефункциональные требования:

- среднее время ответа сервера на запросы, не связанные с выполнением сложных вычислений (получение списка дисциплин, групп, аудиторий), должно составлять не более 500 мс;

- клиентская часть приложения должна корректно отображаться и обеспечивать полноценную функциональность на экранах с разрешением от 1280x720 пикселей и выше (стандартные ноутбуки и мониторы);

- для обеспечения корректной работы интерфейса и выполнения интерактивных функций требуется поддержка браузером следующих технологий:

JavaScript (ES6), HTML5, CSS3 (Flexbox, Grid), Web Storage API (для хранения токенов);

- система должна гарантированно функционировать в последних стабильных версиях следующих браузеров: Google Chrome - мин. версия 90, Mozilla Firefox - мин. версия 88, Microsoft Edge - мин. версия 90, Safari - мин. версия 14, Opera - мин. версия 76;

- для развёртывания системы требуется сервер со следующими минимальными характеристиками: процессор - 4 ядра, 2.5 ГГц, оперативная память - 8 ГБ, дисковое пространство - 20 ГБ (HDD), операционная система - Ubuntu 20.04+, Debian 11+, Windows Server 2019+, СУБД - PostgreSQL 15+.

Ожидаемый результат:

Работоспособное Web-приложение, позволяющее автоматизировать ключевые этапы составления расписания ВУЗа и сократить временные затраты диспетчера и преподавателей на этот процесс, одновременно повысив качество и непротиворечивость итогового документа.

## **2 Выбор технологий и средств разработки приложения**

Выбор технологического стека является критически важным этапом проектирования программной системы масштаба всего высшего учебного заведения. Разрабатываемое Web-приложение составления расписания вуза должно обслуживать сотни преподавателей, десятки учебных подразделений и обеспечивать обработку большого объёма данных (учебные планы, аудиторный фонд, нагрузка) с гарантией согласованности при одновременной работе многих пользователей. В настоящем разделе обосновывается выбор языка программирования серверной части, системы управления базами данных, инструмента администрирования БД и среды разработки с учётом масштаба и требований к надёжности общевузовской информационной системы.

### **2.1 Выбор языка программирования**

Для реализации серверной части разрабатываемого приложения выбран язык Python в сочетании с высокопроизводительным асинхронным Web-

фреймворком FastAPI. Данный выбор обусловлен необходимостью создания масштабируемой, легко поддерживаемой и безопасной системы, способной обрабатывать конкурентные запросы от большого числа пользователей.

Высокая производительность при асинхронной обработке. FastAPI работает поверх ASGI-сервера (Uvicorn) и использует нативный асинхронный цикл событий Python (asyncio). Это позволяет эффективно обслуживать тысячи одновременных соединений при относительно низком потреблении оперативной памяти. Для вузовской системы, где в пиковые периоды (например, начало семестра) одновременно могут работать до 200–300 преподавателей и диспетчеров, асинхронная модель предпочтительнее классической многопоточной (как в Spring Boot или ASP.NET Core) с точки зрения утилизации ресурсов.

Наличие библиотеки для обработки Word-документов. Ключевое требование к системе – импорт поручений из файлов формата .docx, которые в вузе формируются учебным отделом. Библиотека python-docx предоставляет богатый API для чтения таблиц, извлечения текста с учётом форматирования и работы с объединёнными ячейками. Аналоги в других языках либо требуют лицензионных отчислений (Aspose.Words), либо сложнее в использовании (Apache POI). Python-решение является свободным и хорошо документированным.

Быстрота прототипирования и внесения изменений. Учебные планы и структура вуза могут меняться от семестра к семестру. Python с его динамической типизацией и лаконичным синтаксисом позволяет оперативно адаптировать логику парсинга и валидации под новые требования без длительных циклов перекомпиляции, характерных для Java или C#.

Богатая экосистема для научных и образовательных задач. В вузовской среде Python широко используется в учебном процессе и исследовательской работе. Это означает, что при необходимости доработки системы силами ИТ-отдела университета не возникнет проблемы поиска квалифицированных кадров.

Встроенная поддержка современных стандартов безопасности. FastAPI изначально спроектирован с учётом требований OAuth2, JWT и OpenAPI. Автоматическая валидация входных данных через Pydantic и генерация документации Swagger UI упрощают интеграцию с фронтендом и снижают вероятность появления уязвимостей.

Горизонтальная масштабируемость. Благодаря stateless-архитектуре (JWT-токены вместо серверных сессий) экземпляры FastAPI могут быть легко развёрнуты за балансировщиком нагрузки (например, Nginx или Traefik). Это критически важно для обеспечения отказоустойчивости при росте числа пользователей вуза.

Таким образом, выбор Python + FastAPI представляет собой оптимальный вариант, совмещающий производительность, скорость разработки и наличие необходимых для решения специфических задач вуза библиотек.

## **2.2 Выбор СУБД**

В качестве системы управления базами данных для общевузовской системы составления расписания выбрана PostgreSQL. Этот выбор продиктован высокими требованиями к согласованности данных, поддержке конкурентного доступа и способности обрабатывать сложные аналитические запросы в масштабах всего учебного заведения.

Обоснование выбора PostgreSQL для вузовского расписания.

Строгая согласованность данных при конкурентном доступе. В общевузовском расписании над одним и тем же ресурсом (аудитория, группа, преподаватель) одновременно могут работать несколько диспетчеров и преподавателей. Для предотвращения конфликтов необходима поддержка транзакций с уровнем изоляции `SERIALIZABLE`. PostgreSQL реализует этот уровень на основе `Serializable Snapshot Isolation (SSI)`, что гарантирует отсутствие аномалий параллельного выполнения (фантомное чтение, неповторяемое чтение) без чрезмерного использования блокировок. MySQL, напротив, в режиме `SERIALIZABLE` просто блокирует читаемые строки, что резко снижает

конкурентность. Коммерческие СУБД (Oracle, MS SQL) также поддерживают строгую сериализуемость, но их лицензионная стоимость для вуза может быть неприемлемо высокой.

Эффективная работа с JSONB. Поручения и учебные планы содержат множество атрибутов, которые могут варьироваться для разных кафедр и направлений. Тип JSONB позволяет хранить такие переменные данные без изменения схемы базы, при этом поддерживая индексацию (GIN-индексы) для быстрого поиска. Это удобно, например, для хранения индивидуальных ограничений преподавателей, извлекаемых из примечаний Word-файла.

Масштабируемость и отказоустойчивость. PostgreSQL предоставляет встроенные механизмы репликации (потокковая асинхронная и синхронная репликация), что позволяет создать отказоустойчивый кластер с распределением читающей нагрузки между несколькими репликами. Это обеспечивает высокую доступность и производительность.

Интеграция с Python-стеком. Библиотеки `asyncpg` (чистый асинхронный драйвер) и `SQLAlchemy` с поддержкой `asyncio` обеспечивают высокопроизводительное и безопасное взаимодействие с PostgreSQL из FastAPI-приложения. Это позволяет реализовать сложные транзакционные проверки на стороне сервера без блокировки цикла событий.

Отсутствие лицензионных затрат. Для государственного вуза с ограниченным бюджетом возможность использовать мощную СУБД корпоративного класса без оплаты лицензий является существенным преимуществом.

Таким образом, PostgreSQL является СУБД, сочетающей в себе высочайшую надёжность транзакций, гибкость работы с полуструктурированными данными, возможности горизонтального масштабирования и нулевую стоимость владения, что делает её идеальным выбором для общевузовской системы расписания.

## 2.3 Графический клиент pgAdmin 4

Для администрирования базы данных PostgreSQL, мониторинга её состояния, выполнения отладочных запросов и управления структурой данных используется клиент pgAdmin 4. Данный инструмент обладает функциональностью, необходимой для работы с базами данных масштаба вуза.

Обоснование выбора pgAdmin 4.

Полная совместимость с PostgreSQL. pgAdmin 4 разрабатывается PostgreSQL Global Development Group, что гарантирует поддержку всех новейших функций СУБД, включая типы JSONB, политики защиты на уровне строк (RLS), партиционирование таблиц и мониторинг репликации. В условиях вузовской системы, где планируется использовать репликацию для отказоустойчивости, возможность управления и наблюдения за состоянием реплик из единого интерфейса является критически важной.

Web-ориентированная архитектура. pgAdmin 4 может быть развёрнут как Web-приложение на сервере, что позволяет администраторам и разработчикам получать доступ к управлению БД из любой точки сети вуза через браузер без установки клиентского ПО. Это особенно удобно при распределённой структуре университета, где ИТ-специалисты могут находиться в разных корпусах.

Инструменты для анализа производительности. В масштабах вуза база данных расписания будет содержать сотни тысяч записей. pgAdmin 4 включает:

- визуализатор планов выполнения запросов (EXPLAIN), помогающий выявлять медленные запросы и оптимизировать индексы.
- панель мониторинга (Dashboard) с графиками количества транзакций, блокировок, использования дискового пространства и числа подключений.
- журнал сервера с возможностью фильтрации по уровню ошибок.

Эти инструменты незаменимы для поддержания времени отклика системы в пределах установленных нефункциональных требований при росте нагрузки.

Управление пользователями и ролями. В вузовской системе предполагается наличие нескольких ролей на уровне базы данных (роль для преподавателей,

роль для администраторов). pgAdmin 4 предоставляет удобный графический интерфейс для создания ролей, назначения привилегий на схемы, таблицы и даже отдельные строки.

Резервное копирование и восстановление. Встроенные утилиты `pg_dump` и `pg_restore`, интегрированные в интерфейс pgAdmin, позволяют создавать полные или инкрементальные резервные копии критически важных данных расписания. Возможность планирования заданий резервного копирования через pgAgent (входящий в состав pgAdmin) обеспечивает автоматизацию этого процесса без использования сторонних планировщиков.

Бесплатность и кроссплатформенность. Инструмент не требует лицензионных отчислений и может быть запущен на сервере под управлением Linux или Windows.

Альтернативы. Рассматривались универсальные клиенты DBeaver и DataGrip. DBeaver, будучи Java-приложением, потребляет значительно больше оперативной памяти, а DataGrip требует приобретения платной лицензии. pgAdmin 4, являясь специализированным и бесплатным инструментом, представляет собой наиболее рациональный выбор для администрирования PostgreSQL в рамках вузовской информационной системы.

## **2.4 Выбор среды разработки**

Разработка крупной информационной системы для вуза включает одновременную работу над серверной частью на Python (FastAPI) и клиентской частью на React. Для обеспечения единой, гибкой и производительной среды разработки была выбрана VS Code – бесплатный, кроссплатформенный редактор кода с широкими возможностями расширения. Данный выбор обусловлен стремлением унифицировать рабочий процесс, снизить порог входа для будущих разработчиков сопровождения и эффективно использовать ресурсы компьютера при работе с обеими кодовыми базами.

Обоснование выбора Visual Studio Code.

Единая среда для full-stack разработки. Использование одного редактора для backend (Python) и frontend (React) исключает необходимость переключения между разными IDE. Это ускоряет навигацию по проекту, упрощает отладку сквозных сценариев (например, трассировка запроса от UI до БД) и снижает когнитивную нагрузку на разработчика. В рамках вузовского проекта, где кодовая база может сопровождаться разными специалистами, унификация инструмента является значительным преимуществом.

Мощная экосистема расширений. VS Code обладает крупнейшим маркетплейсом расширений, позволяющих превратить редактор в полноценную IDE под конкретные задачи. Для серверной части на Python используются:

- Python (официальное расширение от Microsoft) – поддержка IntelliSense, отладки, тестирования, виртуальных окружений.
- Pylance – быстрый и точный сервер языка для проверки типов и автодополнения, незаменимый при работе с аннотациями типов Pydantic и FastAPI.
- REST Client – выполнение HTTP-запросов непосредственно из редактора для тестирования API.
- Docker – управление контейнерами, просмотр логов, запуск Docker Compose.
- GitLens – расширенная визуализация истории изменений Git.

Для клиентской части на React:

- ES7+ React/Redux/React-Native snippets – быстрая генерация шаблонов компонентов.
- Prettier и ESLint – автоматическое форматирование и линтинг кода.
- Tailwind CSS IntelliSense – автодополнение классов стилей (если используется Tailwind).

Все расширения работают в едином окне, не конфликтуя друг с другом.

Встроенная поддержка отладки для Python и JavaScript. VS Code предоставляет унифицированный интерфейс отладки. Можно одновременно запустить серверную часть (через конфигурацию Python: FastAPI) и клиентскую

(через Chrome: Launch), устанавливать точки останова как в Python-коде (в том числе внутри асинхронных функций), так и в React/JSX-компонентах. Для отладки асинхронного кода FastAPI достаточно добавить конфигурацию с типом `python` и указать `"request": "launch", "module": "uvicorn"`. Это позволяет эффективно исследовать проблемы валидации расписания в реальном времени.

Интеграция с Git и удалёнными репозиториями. VS Code имеет встроенный графический интерфейс для Git: просмотр изменённых файлов, построчное сравнение версий (diff), коммиты, push/pull, разрешение конфликтов слияния. Расширение GitLens добавляет аннотации к каждой строке кода с информацией об авторе и времени последнего изменения, что очень полезно при коллективной разработке и последующем сопровождении системы ИТ-отделом вуза.

Работа с базами данных. Через расширения, такие как PostgreSQL (от Microsoft) или SQLTools, VS Code позволяет подключаться к СУБД PostgreSQL, выполнять произвольные SQL-запросы, просматривать структуру таблиц и даже редактировать данные в графическом виде. Хотя эта функциональность уступает специализированным инструментам вроде pgAdmin, её достаточно для оперативной отладки запросов в процессе написания кода, что ускоряет цикл разработки.

Лёгковесность и низкое потребление ресурсов. По сравнению с полнофункциональными IDE (PyCharm, IntelliJ IDEA) VS Code потребляет значительно меньше оперативной памяти и быстрее запускается. Это особенно важно при работе над крупным проектом с десятками файлов, когда одновременная загрузка двух тяжеловесных IDE для backend и frontend могла бы привести к замедлению работы компьютера разработчика.

Бесплатность и открытость. VS Code распространяется под свободной лицензией MIT и не требует никаких лицензионных отчислений. Это делает его идеальным выбором для государственного вуза с ограниченным ИТ-бюджетом. Более того, любой преподаватель или студент кафедры, который в будущем захочет внести вклад в развитие системы, может бесплатно установить точно такую же среду разработки.

Кроссплатформенность. VS Code работает на Windows, macOS и Linux, что позволяет развернуть среду разработки на любой операционной системе, используемой в университете.

Visual Studio Code в сочетании с правильно подобранным набором расширений полностью покрывает потребности full-stack разработки общеузовской системы составления расписания. Он обеспечивает высокую производительность, удобство отладки, интеграцию с современными инструментами (Git, Docker, БД) и при этом остаётся бесплатным и лёгким. Унификация среды разработки на базе VS Code упрощает командную работу и дальнейшее сопровождение программного продукта силами ИТ-службы университета.

Дополнительные средства разработки:

- Git – система контроля версий, репозиторий размещён на GitHub (или локально).
- Postman – для ручного тестирования API-эндпоинтов (альтернатива – встроенная документация FastAPI /docs).
- Vite – сборщик React-приложения.

### **3 Реализация**

#### **3.1 Архитектура приложения**

Разработанное приложение построено по модели «клиент-сервер» с использованием архитектурного стиля REST. Система состоит из трёх независимо развёртываемых компонентов (рисунок 5):

Серверная часть - реализована на языке Python с применением асинхронного Web-фреймворка FastAPI. Сервер предоставляет REST API для выполнения всех бизнес-операций: аутентификация и авторизация, управление справочными данными (группы, дисциплины, аудитории, преподаватели), работа с элементами расписания, импорт поручений из файлов .docx, экспорт расписания в формат Excel, а также утверждение и отклонение расписания администратором.

Клиентская часть - одностраничное приложение (SPA), разработанное на библиотеке React с использованием компонентного подхода и функциональных хуков. Для построения пользовательского интерфейса применяется библиотека Material UI (MUI), обеспечивающая современный внешний вид и адаптивную вёрстку. Взаимодействие с сервером осуществляется через HTTP-запросы с помощью библиотеки Axios.

База данных - в качестве системы управления базами данных используется PostgreSQL. Взаимодействие серверного приложения с БД организовано через объектно-реляционный маппер SQLAlchemy 2.0 с асинхронным драйвером asyncpg.

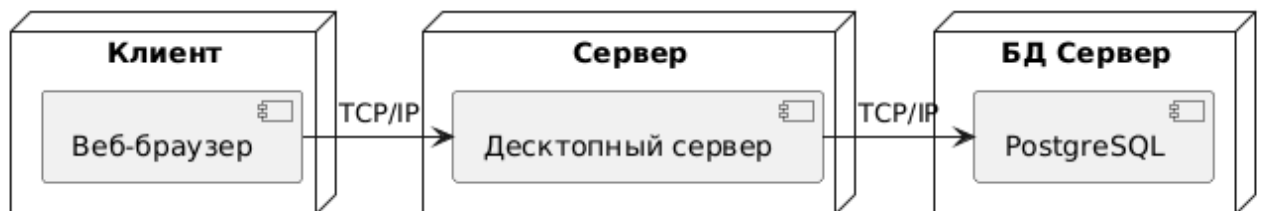


Рисунок 5 - Диаграмма развертывания приложения

Архитектурная схема системы представлена на рисунке 6. Клиентские запросы проходят через слой маршрутизации FastAPI, затем обрабатываются в соответствующих сервисах, которые выполняют валидацию, бизнес-логику и обращаются к базе данных через модели SQLAlchemy. Все критически важные операции (проверка конфликтов при добавлении занятия, отправка на утверждение, утверждение администратором) выполняются в рамках транзакций базы данных для обеспечения согласованности.

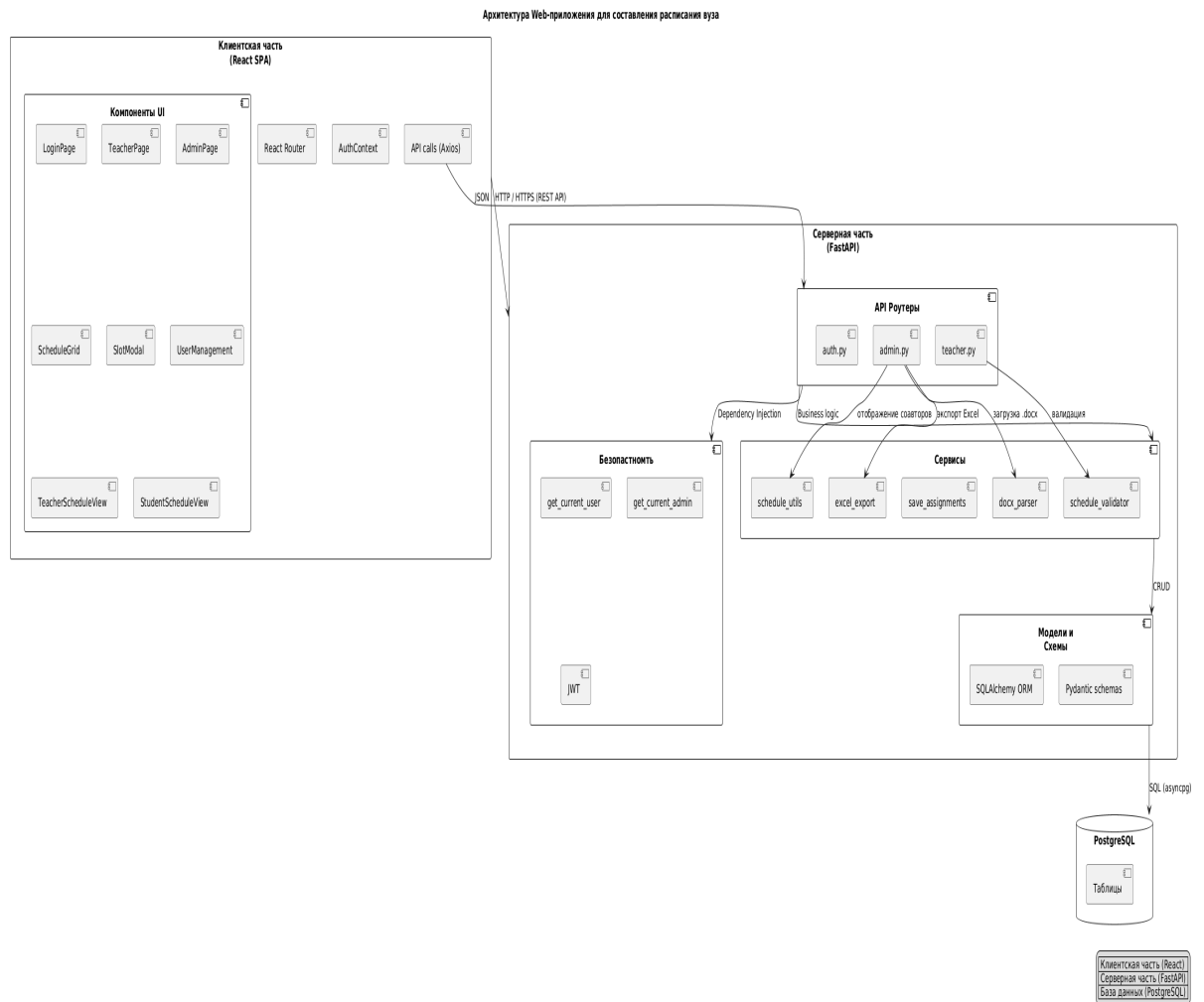


Рисунок 6 – Архитектура приложения

### 3.2 Работа с базой данных

Для хранения информации о расписании спроектирована реляционная база данных. Схема базы данных включает следующие основные сущности (таблицы):

**users** - пользователи системы (преподаватели и администраторы). Содержит поля: идентификатор, логин, хеш пароля, полное имя, роль, учёная степень, кафедра.

**student\_groups** - академические группы. Поля: идентификатор, название, курс, специальность.

**classrooms** - аудитории. Поля: идентификатор, номер, вместимость, наличие проектора, наличие компьютеров, корпус.

disciplines - учебные дисциплины. Поля: идентификатор, название, семестр, часы лекций, часы лабораторных, часы практических.

teaching\_assignments - назначения преподавателей на дисциплины и группы с указанием типа занятий (лекция, лабораторная, практика) и объёма часов. В таблице также присутствует поле secondary\_teacher\_ids (тип JSON), хранящее список идентификаторов преподавателей, приглашённых для ведения совместных лабораторных работ.

schedule\_items - элементы расписания (конкретные занятия). Содержит ссылки на teaching\_assignment, аудиторию, день недели, номер пары, тип недели (все/чётная/нечётная), статус (черновик, на утверждении, утверждён), создателя, примечания, а также поля для реализации механизма совместных занятий: invited\_user\_id (приглашённый преподаватель), collaboration\_master\_id (ссылка на основное занятие-мастер), collaboration\_history (JSON-история сообщений).

Для обеспечения производительности и целостности данных используются индексы по внешним ключам, уникальные ограничения (например, на пару «преподаватель–дисциплина–группа–тип занятия» в teaching\_assignments), а также CHECK-ограничение для номера пары (от 1 до 6).

Создание и обновление схемы базы данных автоматизировано с помощью инструмента миграций Alembic. Все миграции хранятся в каталоге alembic/versions и применяются последовательно при развёртывании приложения.

Для наглядного понимания работы БД, на рисунке 7 представлена схема физической модели базы данных.

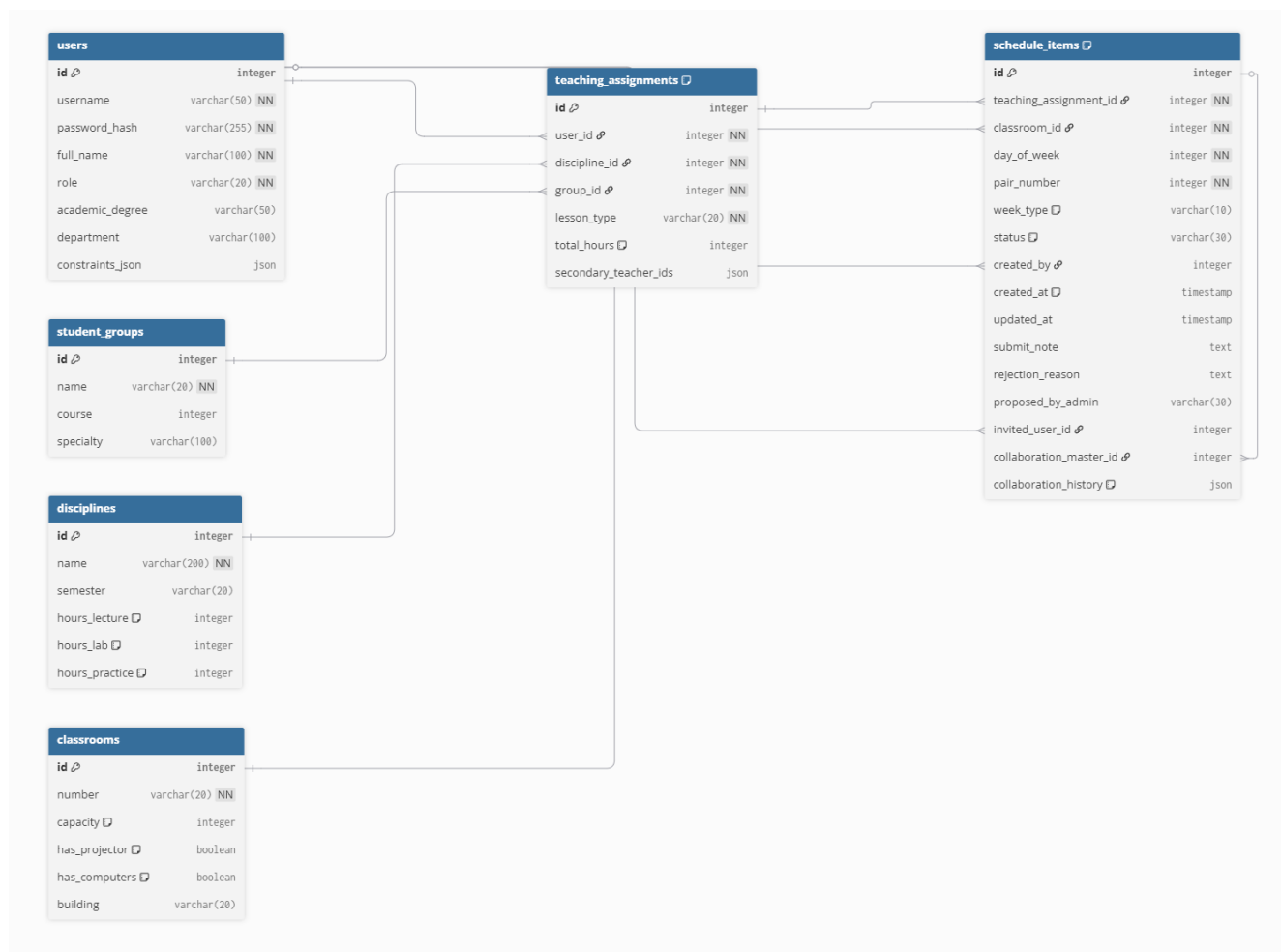


Рисунок 7 – Физическая модель базы данных

### 3.3 Разработка клиентского приложения

Клиентское приложение реализовано в виде одностраничного приложения (SPA) с маршрутизацией на стороне клиента с использованием React Router. Приложение имеет две основные ролевые зоны: администратор и преподаватель. После входа в систему пользователь автоматически перенаправляется на соответствующий его роли интерфейс.

#### 3.3.1 Компонент сетки расписания

Основным компонентом для отображения и редактирования расписания является ScheduleGrid. Он строит таблицу размером 6 дней × 6 пар (для каждой пары отображается ячейка). Каждая ячейка может находиться в одном из состояний:

Свободно (зеленоватый фон) - нет занятия.

Черновик (голубой фон) - занятие создано, но не отправлено на утверждение.

На утверждении (жёлтый фон) - занятие ожидает решения администратора.

Утверждено (розовый фон) - занятие включено в итоговое расписание.

Приглашение (оранжевый фон) - занятие, созданное лектором, ожидает ответа приглашённого преподавателя.

Предложено администратором (фиолетовый фон) - администратор предложил изменения.

Цветовое кодирование позволяет преподавателю и администратору быстро оценить статус каждой позиции. При клике на свободную ячейку открывается форма создания занятия, при клике на черновик - форма редактирования. Для занятий со статусом «утверждено» редактирование недоступно.

### 3.3.2 Модальное окно SlotModal

Компонент SlotModal реализует форму для создания или редактирования элемента расписания. Он динамически загружает список доступных для преподавателя назначений (дисциплина + группа + тип занятия) и, после выбора назначения, запрашивает у сервера список аудиторий, свободных в выбранный день и пару с учётом требований к оборудованию (для лекций - проектор, для лабораторных - компьютеры). Пользователь также может указать тип недели (каждую неделю / только по чётным / только по нечётным).

```
// ScheduleGrid.jsx - фрагмент, определяющий цвет ячейки в зависимости от статуса занятия
const getCellColor = (item) => {
  if (!item) return '#e8f5e9'; // свободно - зелёный
  if (item.status === 'collaboration_waiting') return '#ffe0b2'; //
  приглашение - оранжевый
  if (item.proposed_by_admin) return '#f3e5f5'; //
  предложение админа - фиолетовый
  if (item.status === 'draft') return '#bbdefb'; //
  черновик - голубой
  if (item.status === 'pending') return '#fff9c4'; //
  на утверждении - жёлтый
  if (item.status === 'approved') return '#ffcdd2'; //
  утверждено - розовый
  return '#e0e0e0';
```

```
};
```

Для администратора модальное окно имеет дополнительный режим «предложить правки»: изменения сохраняются со специальным флагом, и преподаватель видит их как предложения.

### 3.3.3 Аутентификация и маршрутизация

Управление состоянием аутентификации реализовано с помощью React Context (AuthContext). При успешном входе токен доступа сохраняется в localStorage и автоматически добавляется в заголовок Authorization каждого HTTP-запроса через Axios. Если сервер возвращает статус 401, токен удаляется, и пользователь перенаправляется на страницу входа.

Аутентификация – интерцептор Axios для добавления токена:

```
// axiosInstance.js
axiosInstance.interceptors.request.use((config) => {
  const token = localStorage.getItem('access_token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Защищённые маршруты реализованы с помощью компонента ProtectedRoute, который проверяет наличие аутентифицированного пользователя и его роль. Это предотвращает несанкционированный доступ к административной панели и преподавательскому интерфейсу.

Диаграмма последовательностей, показывающая процесс входа, изображена на рисунке 8

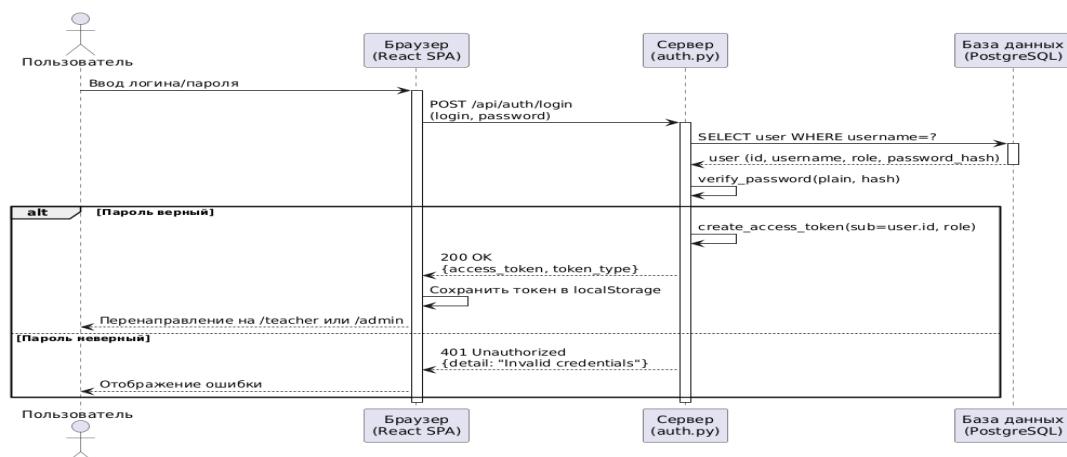


Рисунок 8 - Диаграмма последовательности «Вход»

Процесс создания занятия показан на диаграмме последовательностей (рисунок 9).

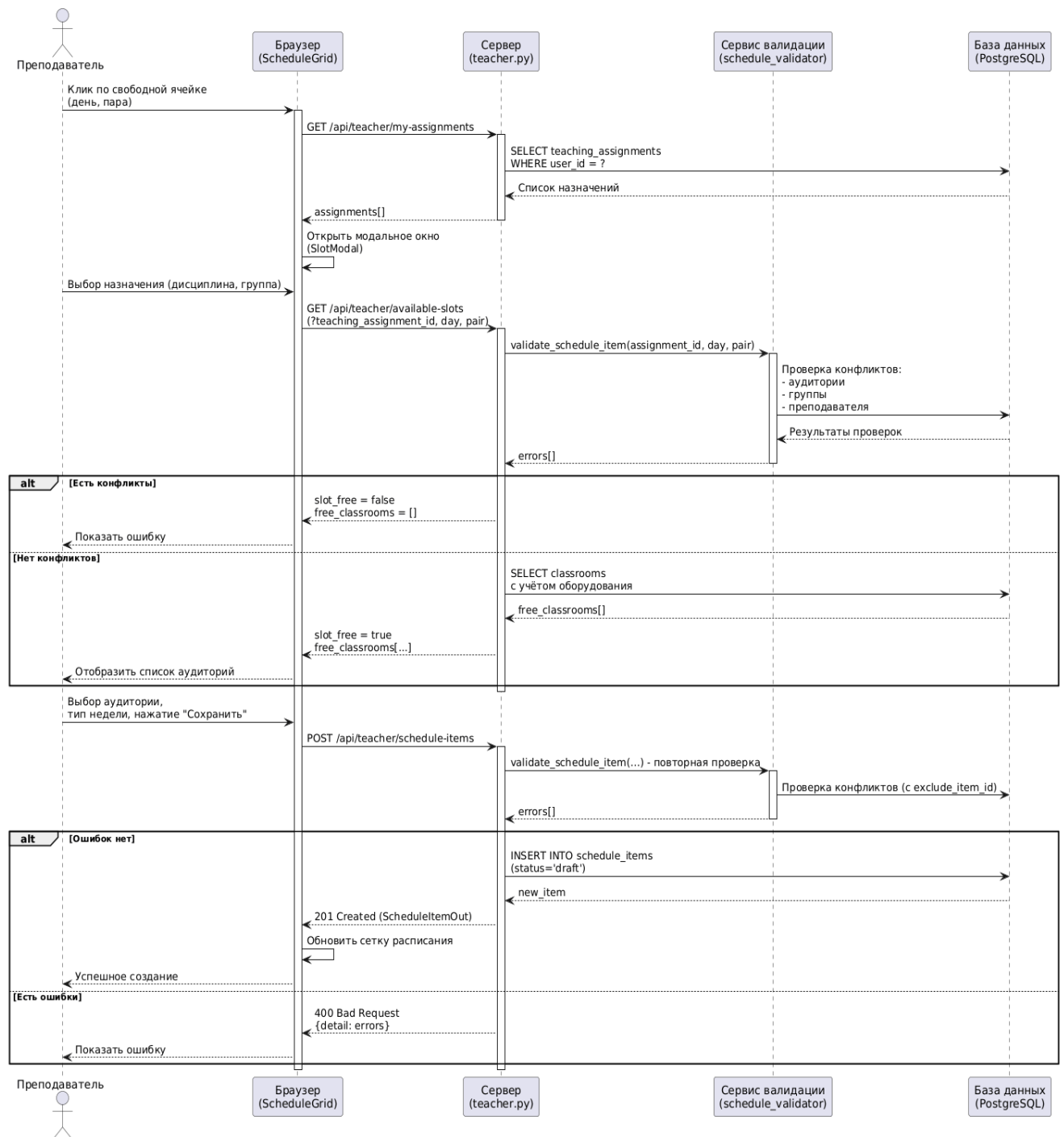


Рисунок 9 - Диаграмма последовательности «Создание занятия»

### 3.4 Разработка серверного приложения

Серверная часть реализована на Python с использованием FastAPI и SQLAlchemy (асинхронный режим).

#### 3.4.1 Модуль аутентификации и авторизации

Аутентификация реализована по протоколу OAuth2 с использованием JWT-токенов. Пароли пользователей хешируются с помощью библиотеки passlib (алгоритм bcrypt). При успешном входе сервер генерирует access-токен с временем жизни 30 минут. Зависимость `get_current_user` извлекает токен из заголовка `Authorization`, верифицирует его и возвращает объект пользователя. Для проверки прав администратора используется зависимость `get_current_admin`, которая вызывает `get_current_user` и дополнительно проверяет роль.

#### 3.4.2 Модуль импорта поручений (`docx_parser`)

Ключевая особенность системы - автоматический импорт учебной нагрузки из файла Microsoft Word, который формируется учебным отделом. Парсер (`app/services/docx_parser.py`) выполняет следующие шаги:

- открывает документ с помощью библиотеки `python-docx`;
- извлекает первую таблицу документа (именно в ней содержится основная информация о дисциплинах, группах, часах и преподавателях);
- последовательно обрабатывает строки таблицы: распознаёт наименование дисциплины, номера групп, количество лекционных и лабораторных часов, фамилии преподавателей (основных и приглашённых), а также текстовые примечания об аудиториях;
- дополнительно парсит раздел «Примечания» после таблицы, чтобы извлечь уточнения об оборудовании аудиторий (наличие проектора, компьютеров);
- из диапазонов групп (например, «25BBB1-4») генерирует отдельные записи для каждой группы;
- из извлечённых данных формирует структурированные списки групп, дисциплин, преподавателей, аудиторий и назначений.

Результат парсинга сохраняется в базу данных через сервис `save_parsed_data`, который избегает дублирования (проверяет существование записей и при необходимости обновляет).

```
# docx_parser.py - функция извлечения групп с поддержкой диапазонов
def _extract_groups_from_text(text: str) -> List[str]:
    groups = []
    group_pattern = re.compile(r'\b(\d{2}[A-ЯЁА-Z]+[\d]+
(?:-\d+)?)\b')
    matches = group_pattern.findall(text)
    for match in matches:
        if '-' in match:
            base, start, end = re.match(r'(.+?) (\d+)-(\d+)$',
match).groups()
            groups.extend([f"{base}{i}" for i in range(int(start),
int(end)+1)])
        else:
            groups.append(match)
    return groups
```

### 3.4.3 Модуль валидации расписания (`schedule_validator`)

Перед сохранением каждого элемента расписания (при создании или редактировании) вызывается асинхронная функция `validate_schedule_item`, которая проверяет:

Занятость аудитории - нет ли другого утверждённого или ожидающего утверждения занятия в этой же аудитории в тот же день и пару.

Занятость группы - нет ли у группы другого занятия в тот же день и пару (с учётом совместных занятий, где одна группа может быть у нескольких преподавателей, но только в рамках одной мастер-записи).

Занятость преподавателя - нет ли у преподавателя другого занятия в это же время.

Дневной лимит группы - не превышает ли количество пар у группы в этот день четырёх.

Оборудование аудитории - если тип занятия «лекция», необходима аудитория с проектором; если «лабораторная» или «практика» - аудитория должна иметь компьютеры.

Все проверки выполняются в рамках одной транзакции, что гарантирует отсутствие состояний гонки при параллельных запросах.

```
# schedule_validator.py - фрагмент проверки конфликта преподавателя
teacher_conflict = (
    select(ScheduleItem)
    .join(TeachingAssignment)
    .where(
        TeachingAssignment.user_id == teacher_id,
        ScheduleItem.day_of_week == day_of_week,
        ScheduleItem.pair_number == pair_number,
        ScheduleItem.status.in_(["pending", "approved", "draft"])
    )
)
if exclude_item_id:
    teacher_conflict = teacher_conflict.where(ScheduleItem.id !=
exclude_item_id)
if (await db.execute(teacher_conflict.limit(1))).scalar():
    errors.append("Преподаватель уже занят в это время")
```

### 3.4.4 Роутеры

Этот модуль один из самых важных, в нём описана логика программы.

auth.py – эндпоинты регистрации, логина, получения информации о текущем пользователе, смены пароля.

teacher.py – все действия преподавателя: получение своих назначений, просмотр и редактирование своего расписания, создание/обновление/удаление занятий, отправка на утверждение, работа с совместными занятиями (принять/отклонить приглашение), экспорт личной ведомости.

admin.py – CRUD-операции для групп, аудиторий, дисциплин; управление пользователями; загрузка поручений; утверждение/отклонение расписаний преподавателей; просмотр и редактирование расписания любого преподавателя; экспорт общей и студенческой ведомостей.

Система поддерживает сценарий, когда лабораторное занятие ведут два преподавателя (лектор и ассистент). В файле поручений такие пары задаются через указание нескольких преподавателей в одной строке. При создании занятия лектором:

Система автоматически определяет, что назначение лабораторного типа имеет поле secondary\_teacher\_ids.

Создаётся мастер-запись ScheduleItem со статусом collaboration\_waiting.

Для каждого вторичного преподавателя создаётся запись-приглашение с тем же `teaching_assignment_id`, но с заполненным полем `invited_user_id` и ссылкой на мастер-запись (`collaboration_master_id`).

Приглашённый преподаватель видит это занятие в своём расписании с синим фоном и может его принять или отклонить.

Принятие приводит к тому, что все связанные записи переходят в статус `draft`, и занятие становится общим для обоих преподавателей. В случае отклонения запускается процедура разделения группы на подгруппы и пересоздания назначений.

```
# teacher.py - создание совместного лабораторного занятия
if is_collaborative:
    schedule_item = ScheduleItem(
        teaching_assignment_id=item.teaching_assignment_id,
        classroom_id=item.classroom_id,
        day_of_week=item.day_of_week,
        pair_number=item.pair_number,
        week_type=item.week_type or "all",
        status="collaboration_waiting",
        created_by=current_user.id,
    )
    db.add(schedule_item)
    await db.flush()
    for s_tid in other_teachers_ids:
        invitation = ScheduleItem(
            teaching_assignment_id=item.teaching_assignment_id,
            classroom_id=item.classroom_id,
            day_of_week=item.day_of_week,
            pair_number=item.pair_number,
            week_type=item.week_type or "all",
            status="draft",
            created_by=current_user.id,
            invited_user_id=s_tid,
            collaboration_master_id=schedule_item.id
        )
        db.add(invitation)
```

### 3.4.5 Экспорт в Excel

Сервер предоставляет два формата экспорта:

Общая ведомость преподавателей - таблица, где строки - преподаватели, столбцы - дни недели, внутри каждого дня - временные слоты. В ячейках отображается группа, дисциплина, тип занятия, преподаватель(и) и аудитория, с пометкой о нечётной/чётной неделе.

Студенческая ведомость - группировка по академическим группам. Для каждой группы строится таблица с разбивкой на нечётные и чётные недели. Используется для выдачи студентам.

Генерация Excel-файлов выполняется с помощью библиотеки `openpyxl`, стилизация (границы, шрифты, объединение ячеек, автоматическая высота строк) приближена к стандартному оформлению, принятому в вузе.

```
# excel_export.py - установка границ и выравнивания
thin_side = Side(style='thin')
thin_border = Border(left=thin_side, right=thin_side, top=thin_side,
bottom=thin_side)
header_fill = PatternFill(start_color="D9D9D9", end_color="D9D9D9",
fill_type="solid")
wrap_alignment = Alignment(wrap_text=True, vertical='top',
horizontal='left')

cell = ws.cell(row=row_num, column=col, value=info)
cell.font = Font(name='Calibri', size=8)
cell.alignment = wrap_alignment
cell.border = thin_border
```

#### 3.4.6 Административные функции

Администратор через API может:

- управлять группами, аудиториями, дисциплинами (CRUD);
- управлять пользователями (создание, редактирование, удаление, просмотр расписания любого преподавателя);
- загружать файл поручений и запускать автоматический импорт;
- просматривать список преподавателей, чьи расписания ожидают утверждения;
- утверждать или отклонять расписание (с возможностью предложить свои правки);
- экспортировать расписание в Excel.

Все административные эндпоинты защищены зависимостью `get_current_admin`.

### 3.5 Тестирование

Были составлены и выполнены формальные тест-кейсы, охватывающие ключевые функции системы. Результаты выполнения представлены в таблице 1.

Таблица 1 - Тест-кейсы функционального тестирования

| Номер теста | Название тест-кейса                 | Шаги                                                                                                          | Ожидаемый результат                                                      | Фактический результат |
|-------------|-------------------------------------|---------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|-----------------------|
| 1           | Вход с корректными учётными данными | Ввести логин admin, пароль admin. Нажать «Войти».                                                             | Перенаправление на страницу /admin.                                      | Выполнено успешно     |
| 2           | Вход с неверным паролем             | Ввести логин admin, неверный пароль.                                                                          | Сообщение об ошибке «Invalid credentials».                               | Выполнено успешно     |
| 3           | Создание занятия преподавателем     | На странице преподавателя кликнуть по свободной ячейке, выбрать дисциплину, аудиторию, тип недели, сохранить. | Занятие появляется в сетке с синим фоном (статус «draft»).               | Выполнено успешно     |
| 4           | Конфликт занятости преподавателя    | Создать два занятия для одного преподавателя в один день и пару.                                              | Второе занятие не создаётся, выводится ошибка «Преподаватель уже занят». | Выполнено успешно     |
| 5           | Конфликт лимита пар у группы        | Создать для одной группы 5 занятий в один день.                                                               | Пятое занятие блокируется, ошибка о превышении 4 пар.                    | Выполнено успешно     |
| 6           | Отправка расписания на утверждение  | Преподаватель нажимает «Отправить на утверждение».                                                            | Все его черновики меняют статус на pending (жёлтый фон).                 | Выполнено успешно     |
| Номер теста | Название тест-кейса                 | Шаги                                                                                                          | Ожидаемый результат                                                      | Фактический результат |
| 7           | Утверждение расписания              | Администратор открывает                                                                                       | Занятия становятся                                                       | Выполнено успешно     |

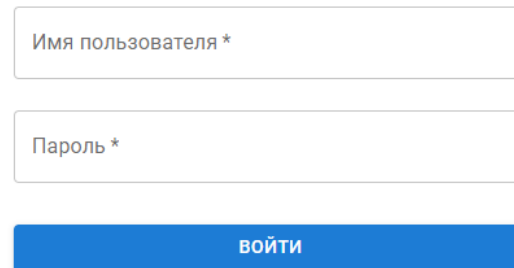
|    |                                                      |                                                                                       |                                                                                                 |                      |
|----|------------------------------------------------------|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|----------------------|
|    | администратор<br>ом                                  | расписание<br>преподавателя,<br>нажимает<br>«Утвердить».                              | approved<br>(розовый фон).                                                                      |                      |
| 8  | Отклонение<br>расписания с<br>предложением<br>правок | Администратор<br>отклоняет<br>расписание с<br>опцией<br>«Предложить<br>исправленное». | Занятия<br>возвращаются в<br>draft, у<br>преподавателя<br>отображаются<br>фиолетовые<br>ячейки. | Выполнено<br>успешно |
| 9  | Принятие<br>совместного<br>приглашения               | Приглашённый<br>преподаватель<br>кликает по<br>занятию,<br>нажимает<br>«Принять».     | Занятие<br>становится<br>общим, у обоих<br>преподавателей<br>отображаются<br>обе фамилии.       | Выполнено<br>успешно |
| 10 | Экспорт<br>преподаватель<br>ской<br>ведомости        | Администратор<br>нажимает<br>«Скачать общую<br>ведомость<br>(Excel)».                 | Скачивается<br>файл<br>schedule.xlsx с<br>корректной<br>структурой.                             | Выполнено<br>успешно |
| 11 | Экспорт<br>студенческой<br>ведомости                 | Администратор<br>нажимает<br>«Скачать<br>студенческую<br>ведомость<br>(Excel)».       | Скачивается<br>файл с<br>группировкой по<br>группам и<br>разбивкой по<br>неделям.               | Выполнено<br>успешно |
| 12 | Создание<br>пользователя<br>администратор<br>ом      | Администратор<br>заполняет форму<br>создания<br>пользователя,<br>сохраняет.           | Новый<br>пользователь<br>появляется в<br>списке.                                                | Выполнено<br>успешно |

В процессе разработки Web-приложения для автоматизированного составления расписания вуза был проведён комплекс мероприятий по тестированию, направленных на проверку корректности работы функциональных требований, производительности, надёжности и безопасности. Тестирование проводилось на всех уровнях: модульное, интеграционное, системное и приёмочное. В качестве сред тестирования использовались фреймворк pytest для серверной части, ручное тестирование API через Postman и инструменты разработчика браузера для клиентской части.

UI-тестирование проводилось вручную в браузерах Google Chrome (версия 90+), Mozilla Firefox (версия 88+) и Microsoft Edge (версия 90+).

Тест 1 - Авторизация и маршрутизация:

Вход в систему составления  
расписания



Имя пользователя \*

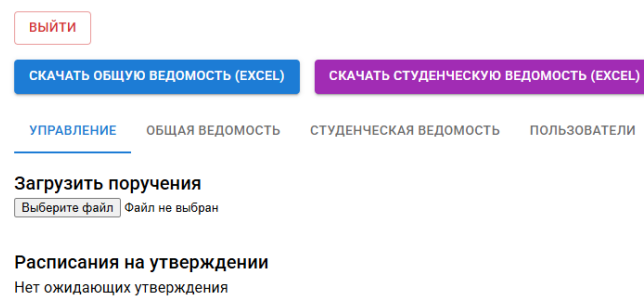
Пароль \*

ВОЙТИ

Рисунок 10 – Страница входа

При вводе корректного логина/пароля пользователь перенаправляется на страницу своей роли (администратор → /admin, преподаватель → /teacher).

Администратор: test11



ВЫЙТИ

СКАЧАТЬ ОБЩУЮ ВЕДОМОСТЬ (EXCEL) СКАЧАТЬ СТУДЕНЧЕСКУЮ ВЕДОМОСТЬ (EXCEL)

УПРАВЛЕНИЕ ОБЩАЯ ВЕДОМОСТЬ СТУДЕНЧЕСКАЯ ВЕДОМОСТЬ ПОЛЬЗОВАТЕЛИ

Загрузить поручения

Выберите файл | Файл не выбран

Расписания на утверждении

Нет ожидающих утверждения

Рисунок 11 – Страница администратора

Добро пожаловать, Акифьев И.В.!

РЕДАКТИРОВАНИЕ

ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ

ВЫЙТИ

Примечание к отправке

Итого  
1-я неделя (9 февр. – 14 февр.)

|   | Пн10 окт                                                                                         | Вт10 окт                                                                                         | Ср11 окт | Чт12 окт | Пт13 окт | Сб14 окт |
|---|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|----------|----------|----------|----------|
| 1 | Декларативные языки программирования<br>248881.12 (им.)<br>Акифьев И.В.<br>кв. 7а-317            | Свободно                                                                                         | Свободно | Свободно | Свободно | Свободно |
| 2 | Декларативные языки программирования<br>248884 (им.)<br>Дубинин В.Н., Акифьев И.В.<br>кв. 7а-317 | Декларативные языки программирования<br>248884 (им.)<br>Дубинин В.Н., Акифьев И.В.<br>кв. 7а-317 | Свободно | Свободно | Свободно | Свободно |
| 3 | Декларативные языки программирования<br>248884 (им.)<br>Дубинин В.Н., Акифьев И.В.<br>кв. 7а-317 | Свободно                                                                                         | Свободно | Свободно | Свободно | Свободно |
| 4 | Свободно                                                                                         | Свободно                                                                                         | Свободно | Свободно | Свободно | Свободно |
| 5 | Свободно                                                                                         | Свободно                                                                                         | Свободно | Свободно | Свободно | Свободно |
| 6 | Свободно                                                                                         | Свободно                                                                                         | Свободно | Свободно | Свободно | Свободно |

Свободно

Черновик

На утверждении

Утверждено

Приглашение адм.

Предложено

## Рисунок 12 – Страница преподавателя

При отсутствии токена попытка доступа к защищённым маршрутам перенаправляет на /login.

Нажатие кнопки «Выйти» → удаление токена, редирект на /login.

Тест 2 - Просмотр расписания и создание занятия:

Корректное отображение занятий по дням и парам.

Цветовая индикация в соответствии со статусами (черновик – синий, на утверждении – жёлтый, утверждён – красный, приглашение – оранжевый, предложение администратора – фиолетовый, свободный слот - зелёный).

Клик по свободной ячейке открывает модальное окно создания занятия.

Добавить занятие: Вт, 1-я пара

Дисциплина / группа

▼

Аудитория

▼

Тип недели

Каждую неделю

▼

СОЗДАТЬ

## Рисунок 13 – Модальное окно создания занятия

Список назначений (дисциплина + группа + тип) загружается корректно.

|                                                                            |
|----------------------------------------------------------------------------|
| Информационные технологии в профессиональной деятельности (25BA1, lab)     |
| Информационные технологии в профессиональной деятельности (25BBB4, lab)    |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB1, lecture) |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB2, lecture) |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB3, lecture) |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB4, lecture) |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ЭЭ1, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ЭЭ2, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB1, lab)     |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB2, lab)     |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB3, lab)     |
| Инсталляция и эксплуатация вычислительных систем и сетей (24BBB4, lab)     |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВЭ1, lab)      |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВЭ2, lab)      |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВИ1, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВИ2, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВП1, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВП2, lecture)  |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВИ1, lab)      |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВИ2, lab)      |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВП1, lab)      |
| Инсталляция и эксплуатация вычислительных систем и сетей (24ВП2, lab)      |
| Декларативные языки программирования (24BBB1/2, lab)                       |
| Декларативные языки программирования (24BBB2/2, lab)                       |
| Декларативные языки программирования (24BBB3/2, lab)                       |
| Декларативные языки программирования (24BBB4, lab)                         |

Рисунок 14 – Список назначений

После выбора назначения автоматически загружается список свободных аудиторий (с учётом оборудования: для лекции - проектор, для лаб. - компьютеры).

Добавить занятие: Вт, 1-я пара

Дисциплина / группа  
Информационные технологии в профессиональной деятельнос...

Аудитория

- 76-204 есть проектор, есть компьютер
- 7а-317 есть компьютер
- 7а-320 есть проектор, есть компьютер
- 7а-319 есть проектор, есть компьютер
- 7а-322 есть компьютер
- 76-202 есть проектор, есть компьютер
- 7а-316 есть проектор, есть компьютер

Рисунок 15 – Список свободных аудиторий

Выбор аудитории, типа недели (все/чётная/нечётная). При сохранении без конфликтов занятие появляется в сетке (синим цветом).

Добро пожаловать, Гурин Е.И.!

РЕДАКТИРОВАНИЕ ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ ВЫЙТИ Примечание к отправке Неделя 1-я неделя (9 февр. – 14 февр.) ▼

|   | Пн9 февр. | Вт10 февр.                                                                                                              | Ср11 февр. | Чт12 февр. |
|---|-----------|-------------------------------------------------------------------------------------------------------------------------|------------|------------|
| 1 | Свободно  | Проектирование устройств на программируемых логических интегральных схемах 22ВВВ1 (лекция)<br>Гурин Е.И.<br>ауд. 76-204 | Свободно   | Свободно   |

Рисунок 16 – Черновик

Добро пожаловать, Карамышева Н.С.!

РЕДАКТИРОВАНИЕ ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ ВЫЙТИ Примечание к отправке Неделя 1-я неделя (9 февр. – 14 февр.) ▼

|   | Пн9 февр.                                                            | Вт10 февр. | Ср11 февр. | Чт12 февр. |
|---|----------------------------------------------------------------------|------------|------------|------------|
| 1 | Web-программирование 22ВВВ1 (лаб.)<br>Карамышева Н.С.<br>ауд. 7а-317 | Свободно   | Свободно   | Свободно   |

Рисунок 17 – Отправленно на утверждение

При конфликте (преподаватель занят, группа занята, превышение 4 пар в день) → сообщение об ошибке, занятие не создаётся Занятые аудитории не показываются в списке.

Добавить занятие: Пн, 1-я пара

❗ Группа уже имеет занятие в это время

Дисциплина / группа  
Проектирование устройств на программируемых логических ин... ▼

Аудитория  
7а-320 есть проектор, есть компьютер ▼

Тип недели  
Каждую неделю ▼

СОЗДАТЬ

Рисунок 18 – Конфликт

Клик по занятию в статусе «черновик» открывает окно редактирования; занятие в статусе «утверждён» не редактируется.

Редактирование: Проектирование устройств на программируемых логических интегральных схемах

Дисциплина / группа  
Проектирование устройств на программируемых логических ин... ▼

Аудитория  
76-204 есть проектор, ▼

Тип недели  
Каждую неделю ▼

**СОХРАНИТЬ ИЗМЕНЕНИЯ** **УДАЛИТЬ**

Рисунок 19 – Редактирование

Отправка формы приводит к обновлению сетки расписания без перезагрузки страницы.

Тест 3 - Совместные занятия (коллаборация).

Создание совместного занятия (лектором).

Если у назначения (teaching\_assignment) заполнено поле secondary\_teacher\_ids, создаётся мастер-запись и приглашения.

У лектора занятие отображается с оранжевым фоном (collaboration\_waiting).

Добро пожаловать, Дубинин В.Н.!

РЕДАКТИРОВАНИЕ

ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ

ВЫЙТИ

Примечание к отправке

Неделя

1-я неделя (9 февр. – 14 февр.)

|   | Пн9 февр.                                                                                                                             | Вт10 февр. | Ср11 февр. | Чт12 февр. |
|---|---------------------------------------------------------------------------------------------------------------------------------------|------------|------------|------------|
| 1 | Свободно                                                                                                                              | Свободно   | Свободно   | Свободно   |
| 2 | <div>Декларативные языки программирования</div> <div>248ВВ1 (лаб.)</div> <div>Дубинин В.Н., Акифьев И.В.</div> <div>ауд. 7а-317</div> | Свободно   | Свободно   | Свободно   |

Рисунок 20 – Создание совместного занятия

У приглашённого преподавателя занятие появляется в сетке с синим фоном.

Добро пожаловать, Акифьев И.В.!

РЕДАКТИРОВАНИЕ    ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ    ВЫЙТИ    Примечание к отправке    Неделя 1-я неделя (9 ф

|   | Пн9 февр.                                                                                          | Вт10 февр. | Ср11 февр. |
|---|----------------------------------------------------------------------------------------------------|------------|------------|
| 1 | Свободно                                                                                           | Свободно   | Свободно   |
| 2 | Декларативные языки программирования<br>24BBB1 (лаб.)<br>Дубинин В.Н., Акифьев И.В.<br>ауд. 7а-317 | Свободно   | Свободно   |

Рисунок 21 – Совместное занятия

Принятие приглашения (приглашённый преподаватель)

Клик по синему приглашению → открывается модальное окно с кнопками «Принять» / «Отклонить».

Редактирование: Декларативные языки программирования

Дисциплина / группа  
Декларативные языки программирования (24BBB1)

Аудитория  
7а-317

Тип недели  
Каждую неделю

ПРИНЯТЬ    ОТКЛОНИТЬ

Рисунок 22 – Приглашение

Принятие → у обоих преподавателей занятие становится синим (draft). Это даёт возможность отправить занятие на утверждение администратору. В ячейке у обоих отображаются фамилии обоих преподавателей. При утверждении администратором – у обоих становится красным (approved).

Отклонение приглашения

Клик «Отклонить» → возможность указать причину.

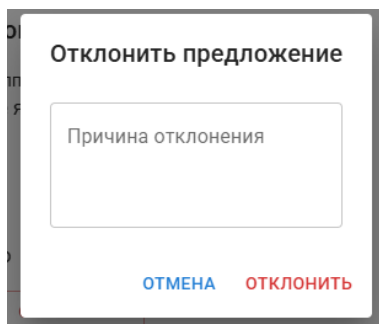


Рисунок 23 – Отклонение предложения

Занятие удаляется у обоих и группа разделяется на 2 подгруппы. Каждый преподаватель теперь работает со своей подгруппой.

инсталляции и эксплуатации вычислительных систем и сетей (24BBB1/2, lab)  
Декларативные языки программирования (24BBB1/2, lab)  
Инсталляция и эксплуатация вычислительных систем и сетей (24BBB1/2, lab)

Рисунок 24 – Разделение группы

Тест 4 – Утверждение администратором.

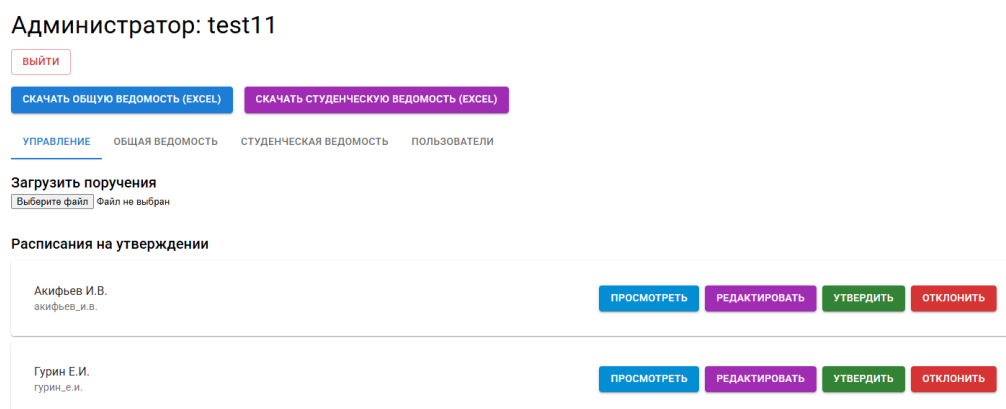


Рисунок 25 – Панель администратора с заявками

Просмотр списка преподавателей с расписанием на утверждении. В разделе «Управление» отображаются преподаватели, у которых есть занятия со статусом pending.

Просмотр расписания преподавателя. Кнопка «Просмотреть» → открывается диалог с сеткой расписания.

Кнопка «Редактировать» → открывается диалог с возможностью изменения (администратор может добавлять, редактировать, удалять занятия).

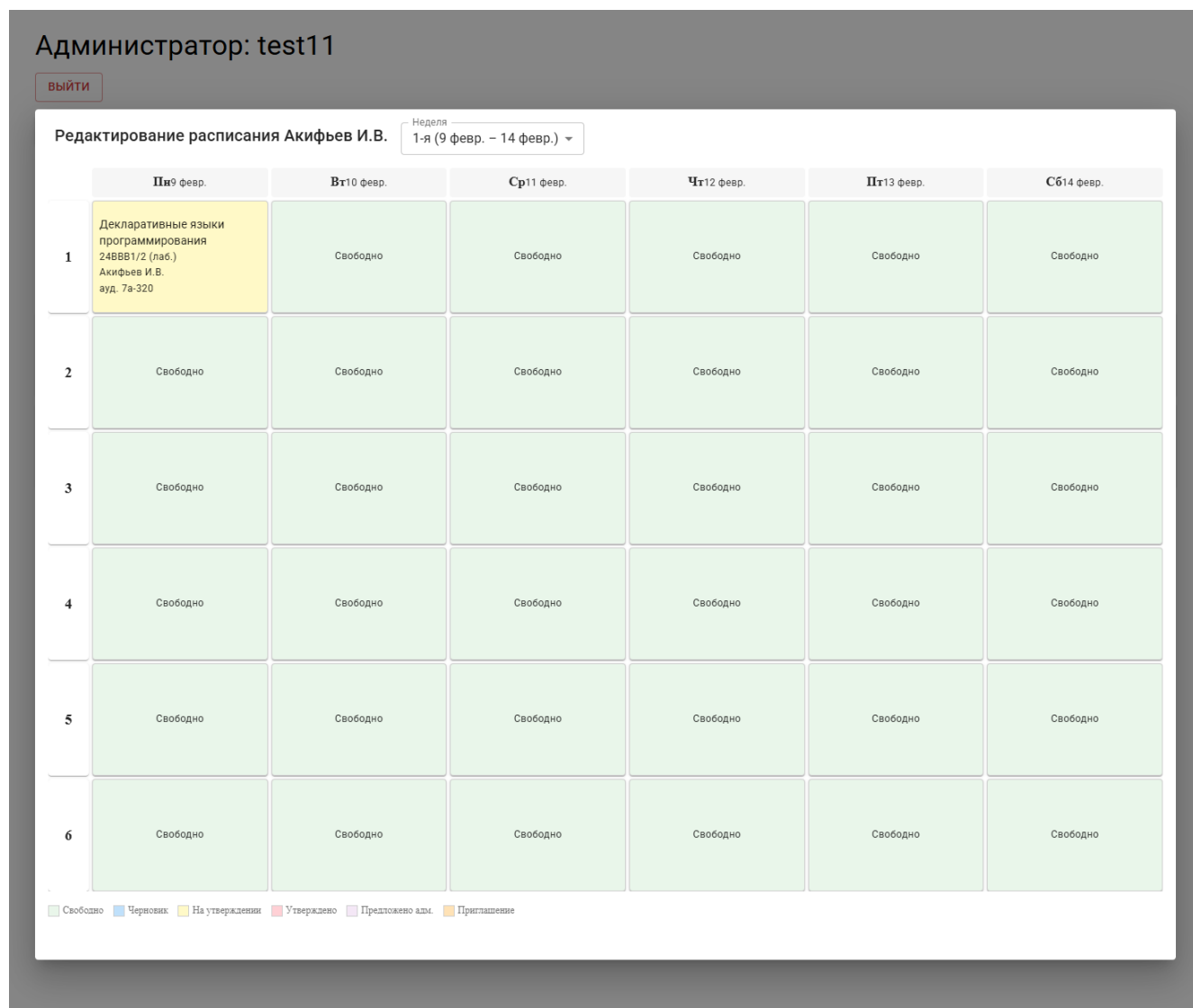


Рисунок 26 – Диалог с сеткой расписания

Редактирование занятия администратором. Клик по ячейке (свободной или занятой) → открывается модальное окно, аналогичное преподавательскому. Можно выбрать любое назначение преподавателя, аудиторию.

Утверждение расписания. Кнопка «Утвердить» → все pending занятия этого преподавателя становятся approved (синхронно для совместных).

Отклонение расписания. Кнопка «Отклонить» → открывается диалог с указанием причины и опцией «Предложить исправленное расписание».

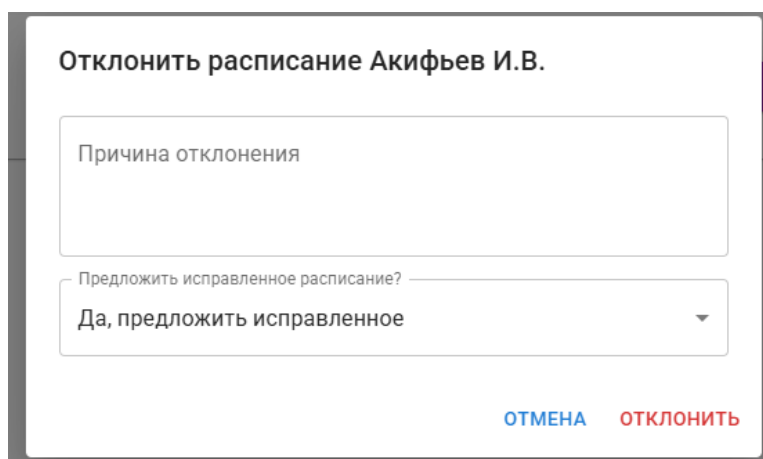
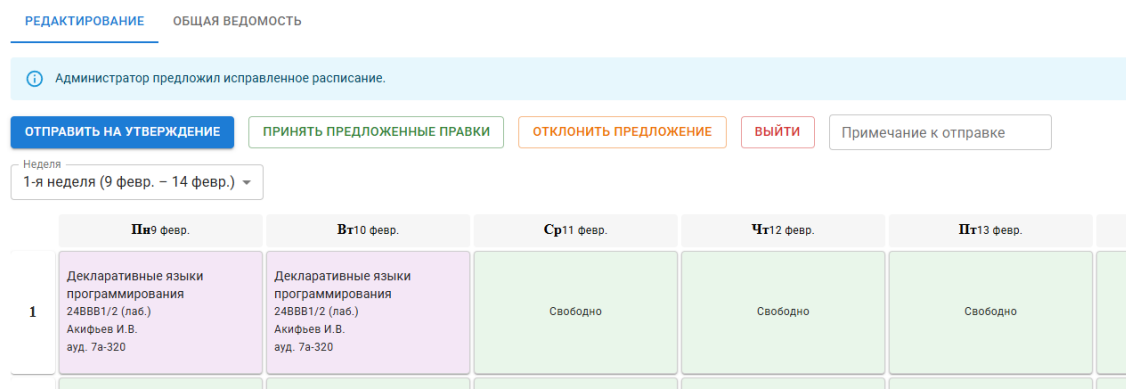


Рисунок 27 – Отклонение расписания

Если выбран пункт «Предложить исправленное» → администратор может отредактировать расписание, и преподаватель увидит фиолетовые ячейки.

Добро пожаловать, Акифьев И.В.!



|   | Пн9 февр.                                                                              | Вт10 февр.                                                                             | Ср11 февр. | Чт12 февр. | Пт13 февр. |
|---|----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|------------|------------|------------|
| 1 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Свободно   | Свободно   | Свободно   |

Рисунок 28 – Предложения администратора

При отклонении, занятия не утверждаются и преподаватель может изменить их или создать новые и снова отправить на утверждение администратору.

# Добро пожаловать, Акифьев И.В.!

РЕДАКТИРОВАНИЕ

ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ

ВЫЙТИ

Примечание к отправке

Неделя  
1-я неделя (9 с

|   | Пн9 февр.                                                                              | Вт10 февр.                                                                             | Ср11 февр. |
|---|----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|------------|
| 1 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Свободно   |

Рисунок 29 – Отклонение предложения

При принятии занятия автоматически отправляются на утверждение.

# Добро пожаловать, Акифьев И.В.!

РЕДАКТИРОВАНИЕ

ОБЩАЯ ВЕДОМОСТЬ

ОТПРАВИТЬ НА УТВЕРЖДЕНИЕ

ВЫЙТИ

Примечание к отправке

Неделя  
1-я неделя (9 с

|   | Пн9 февр.                                                                              | Вт10 февр.                                                                             | Ср11 февр. |
|---|----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|------------|
| 1 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Декларативные языки программирования<br>248ВВ1/2 (лаб.)<br>Акифьев И.В.<br>ауд. 7а-320 | Свободно   |

Рисунок 30 – Принятие предложения

Тест 5 – Работа с пользователями.

Просмотр списка всех пользователей.

Администратор: test11

Выйти

СКАЧАТЬ ОБЩУЮ ВЕДОМОСТЬ (EXCEL)СКАЧАТЬ СТУДЕНЧЕСКУЮ ВЕДОМОСТЬ (EXCEL)

УПРАВЛЕНИЕ

ОБЩАЯ ВЕДОМОСТЬ

СТУДЕНЧЕСКАЯ ВЕДОМОСТЬ

ПОЛЬЗОВАТЕЛИ

ДОБАВИТЬ ПОЛЬЗОВАТЕЛЯ

| ID  | Логин             | ФИО               | Роль    | Учёная степень | Кафедра    | Действия                  |
|-----|-------------------|-------------------|---------|----------------|------------|---------------------------|
| 1   | test1             | test11            | admin   |                |            | ИЗМЕНИТЬ                  |
| 277 | акифьев_и.в.      | Акифьев И.В.      | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 278 | барышева_т.а.     | Барышева Т.А.     | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 279 | бикташев_р.а.     | Бикташев Р.А.     | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 280 | бычков_а.с.       | Бычков А.С.       | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 281 | генералова_а.а.   | Генералова А.А.   | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 282 | горбунов_в.н.     | Горбунов В.Н.     | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 283 | гурин_е.и.        | Гурин Е.И.        | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 284 | деев_м.в.         | Деев М.В.         | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 285 | дубинин_в.н.      | Дубинин В.Н.      | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 286 | забродина_к.а.    | Забродина К.А.    | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 287 | зинкин_с.а.       | Зинкин С.А.       | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 288 | исхаков_н.в.      | Исхаков Н.В.      | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 289 | калининченко_е.и. | Калининченко Е.И. | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 290 | карамышева_н.с.   | Карамышева Н.С.   | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 291 | кузнецов_а.в.     | Кузнецов А.В.     | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 292 | леонова_т.ю.      | Леонова Т.Ю.      | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 293 | митрохин_м.а.     | Митрохин М.А.     | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 294 | митрохина_н.ю.    | Митрохина Н.Ю.    | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 295 | никишин_к.и.      | Никишин К.И.      | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |
| 296 | панков_а.а.       | Панков А.А.       | teacher |                | Кафедра ВТ | ИЗМЕНИТЬРАСПИСАНИЕУДАЛИТЬ |

Рисунок 31 – Список пользователей

Создание нового пользователя (задание роли teacher/admin).

**Создать пользователя**

Логин

Пароль (оставьте пустым, чтобы не менять)

ФИО

Роль  
Преподаватель

Учёная степень

Кафедра

ОТМЕНА СОХРАНИТЬ

Рисунок 32 – Создание пользователя

| ID  | Логин | ФИО | Роль    | Учёная степень | Кафедра | Действия                                                                    |
|-----|-------|-----|---------|----------------|---------|-----------------------------------------------------------------------------|
| 307 | 123   | 123 | admin   |                |         | <a href="#">ИЗМЕНИТЬ</a>                                                    |
| 308 | йцу   | йцу | teacher |                |         | <a href="#">ИЗМЕНИТЬ</a> <a href="#">РАСПИСАНИЕ</a> <a href="#">УДАЛИТЬ</a> |

Рисунок 33 – Новый пользователь

Редактирование пользователя (смена роли, ФИО, сброс пароля).

**Редактировать пользователя**

Логин  
барышева\_т.а.

Пароль (оставьте пустым, чтобы не менять)

ФИО  
Барышева Т.А.

Роль  
Преподаватель

Учёная степень

Кафедра  
Кафедра ВТ

ОТМЕНА СОХРАНИТЬ

Рисунок 34 – Редактирование пользователя

Удаление пользователя (с проверкой: нельзя удалить администратора).

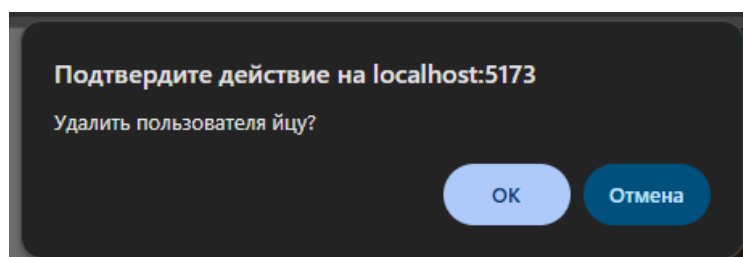


Рисунок 35 – Удаление пользователя

Загрузка файла .docx

Выбор файла, нажатие «Загрузить». Повторная загрузка того же файла не создаёт дубликатов (обновляет существующие записи). При ошибках парсинга (неправильный формат) → ничего не добавляется. Проверка поддержки диапазонов групп (например, «25BBB1-4» → создаются 4 группы).

Тест 6 - Экспорт расписания.

Экспорт общей ведомости преподавателей. Кнопка «Скачать общую ведомость (Excel)» → скачивается файл с корректной структурой:

- строки – преподаватели, столбцы – дни × пары, в ячейках – группа, дисциплина, тип, аудитория, пометка недели.

| № п/п | Ф.И.О.          | Понедельник                                                                               |            |                                                                                                     |             |             |             |             | Вторник                                                                                                                           |                                                                                                          |             |             |             |             |             |
|-------|-----------------|-------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------|-------------|-------------|-------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|-------------|-------------|-------------|-------------|-------------|
|       |                 | 8.00-9.35                                                                                 | 9.50-11.25 | 11.40-13.15                                                                                         | 13.45-15.20 | 15.35-17.10 | 17.25-19.00 | 19.15-20.50 | 8.00-9.35                                                                                                                         | 9.50-11.25                                                                                               | 11.40-13.15 | 13.45-15.20 | 15.35-17.10 | 17.25-19.00 | 19.15-20.50 |
| 1     | Дубинин В.Н.    |                                                                                           |            | 248881/1<br>Декларативные<br>языки<br>программирован<br>ия лб. Дубинин<br>В.Н. ауд.7а-320<br>(чет.) |             |             |             |             |                                                                                                                                   | 248882<br>Декларативные<br>языки<br>программирован<br>ия лб. Дубинин<br>В.Н., Акифеев И.В.<br>ауд.7а-320 |             |             |             |             |             |
| 2     | Акифеев И.В.    | 248881/2<br>Декларативные<br>языки<br>программирован<br>ия лб. Акифеев<br>И.В. ауд.7а-320 |            |                                                                                                     |             |             |             |             | 248881/2<br>Декларативные<br>языки<br>программирован<br>ия лб. Акифеев<br>И.В. ауд.7а-320                                         |                                                                                                          |             |             |             |             |             |
| 3     | Карамышева Н.С. | 228881 Web-<br>программирован<br>ие лб.<br>Карамышева Н.С.<br>ауд.7а-317                  |            |                                                                                                     |             |             |             |             |                                                                                                                                   |                                                                                                          |             |             |             |             |             |
| 4     | Гурин Е.И.      |                                                                                           |            |                                                                                                     |             |             |             |             | 228881<br>Проектирование<br>устройств на<br>программируемы<br>х логических<br>интегральных<br>схемах лб. Гурин<br>Е.И. ауд.7б-204 |                                                                                                          |             |             |             |             |             |

Рисунок 36 – Преподавательское расписание

Для совместных занятий в ячейке перечислены оба преподавателя.

Экспорт студенческой ведомости. Кнопка «Скачать студенческую ведомость (Excel)» → скачивается файл с группировкой по группам, разбивкой на чётные/нечётные недели.

| дни | часы        | 22BBB1                                                                                                     | дни | часы        | 24BBB1/1                                                               | дни | часы        | 24BBB1/2                                                               | дни | часы        | 24BBB2                                                                               |
|-----|-------------|------------------------------------------------------------------------------------------------------------|-----|-------------|------------------------------------------------------------------------|-----|-------------|------------------------------------------------------------------------|-----|-------------|--------------------------------------------------------------------------------------|
| ПН  | 8:00-9:35   | Web-программирование (лаб.)<br>Карамышева Н.С. ауд.7а-317                                                  | ПН  | 8:00-9:35   |                                                                        | ПН  | 8:00-9:35   | Декларативные языки программирования (лаб.)<br>Акифьев И.В. ауд.7а-320 | ПН  | 8:00-9:35   |                                                                                      |
|     | 9:50-11:25  |                                                                                                            |     | 9:50-11:25  |                                                                        |     | 9:50-11:25  |                                                                        |     | 9:50-11:25  |                                                                                      |
|     | 11:40-13:15 |                                                                                                            |     | 11:40-13:15 | Декларативные языки программирования (лаб.)<br>Дубинин В.Н. ауд.7а-320 |     | 11:40-13:15 |                                                                        |     | 11:40-13:15 |                                                                                      |
|     | 13:45-15:20 |                                                                                                            |     | 13:45-15:20 |                                                                        |     | 13:45-15:20 |                                                                        |     | 13:45-15:20 |                                                                                      |
|     | 15:35-17:10 |                                                                                                            |     | 15:35-17:10 |                                                                        |     | 15:35-17:10 |                                                                        |     | 15:35-17:10 |                                                                                      |
|     | 17:25-19:00 |                                                                                                            |     | 17:25-19:00 |                                                                        |     | 17:25-19:00 |                                                                        |     | 17:25-19:00 |                                                                                      |
|     | 19:15-20:50 |                                                                                                            |     | 19:15-20:50 |                                                                        |     | 19:15-20:50 |                                                                        |     | 19:15-20:50 |                                                                                      |
| ВТ  | 8:00-9:35   | Проектирование устройств на программируемых логических интегральных схемах (лек.)<br>Гурин Е.И. ауд.7б-204 | ВТ  | 8:00-9:35   |                                                                        | ВТ  | 8:00-9:35   | Декларативные языки программирования (лаб.)<br>Акифьев И.В. ауд.7а-320 | ВТ  | 8:00-9:35   |                                                                                      |
|     | 9:50-11:25  |                                                                                                            |     | 9:50-11:25  |                                                                        |     | 9:50-11:25  |                                                                        |     | 9:50-11:25  |                                                                                      |
|     | 11:40-13:15 |                                                                                                            |     | 11:40-13:15 |                                                                        |     | 11:40-13:15 |                                                                        |     | 11:40-13:15 | Декларативные языки программирования (лаб.)<br>Дубинин В.Н., Акифьев И.В. ауд.7а-320 |

Рисунок 37 – Студенческое расписание

Тест 7 – Просмотр расписания онлайн.

Во вкладке «Общая ведомость» должно быть расписание всех преподавателей (эта вкладка также должна быть у преподавателей).

| № | Ф.И.О.          | понедельник                                                                               |            |                                                                                                  |             |             |             |             | вторник                                                                                                                           |            |                                                                                                          |             |             |             |             |
|---|-----------------|-------------------------------------------------------------------------------------------|------------|--------------------------------------------------------------------------------------------------|-------------|-------------|-------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------|-------------|-------------|-------------|-------------|
|   |                 | 8.00-9.35                                                                                 | 9.50-11.25 | 11.40-13.15                                                                                      | 13.45-15.20 | 15.35-17.10 | 17.25-19.00 | 19.15-20.50 | 8.00-9.35                                                                                                                         | 9.50-11.25 | 11.40-13.15                                                                                              | 13.45-15.20 | 15.35-17.10 | 17.25-19.00 | 19.15-20.50 |
| 1 | Дубинин В.Н.    |                                                                                           |            | 24BBB1/1<br>Декларативные<br>языки<br>программирования<br>лб. Дубинин В.Н.<br>ауд. 7а-320 (чет.) |             |             |             |             |                                                                                                                                   |            | 24BBB2<br>Декларативные<br>языки<br>программирования<br>лб. Дубинин В.Н.,<br>Акифьев И.В.<br>ауд. 7а-320 |             |             |             |             |
| 2 | Акифьев И.В.    | 24BBB1/2<br>Декларативные<br>языки<br>программирования<br>лб. Акифьев И.В.<br>ауд. 7а-320 |            |                                                                                                  |             |             |             |             | 24BBB1/2<br>Декларативные<br>языки<br>программирования<br>лб. Акифьев И.В.<br>ауд. 7а-320                                         |            |                                                                                                          |             |             |             |             |
| 3 | Карамышева Н.С. | 22BBB1 Web-<br>программирование<br>лб. Карамышева<br>Н.С. ауд. 7а-317                     |            |                                                                                                  |             |             |             |             |                                                                                                                                   |            |                                                                                                          |             |             |             |             |
| 4 | Гурин Е.И.      |                                                                                           |            |                                                                                                  |             |             |             |             | 22BBB1<br>Проектирование<br>устройств на<br>программируемых<br>логических<br>интегральных<br>схемах лб. Гурин<br>Е.И. ауд. 7б-204 |            |                                                                                                          |             |             |             |             |

Рисунок 38 – Общая ведомость

Во вкладке «Студенческая ведомость» должно быть расписание для каждой группы у которых есть хотя-бы 1 пара.

| дни | часы        | 22BBB1                                                                                                         | 24BBB1/1                                                                   | 24BBB1/2                                                                   | 24BBB2                                                                                   |
|-----|-------------|----------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|----------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| ПН  | 8:00-9:35   | Web-программирование (лаб.)<br>Карамышева Н.С. ауд. 7а-317                                                     |                                                                            | Декларативные языки программирования<br>(лаб.)<br>Акифьев И.В. ауд. 7а-320 |                                                                                          |
|     | 9:50-11:25  |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 11:40-13:15 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 13:45-15:20 |                                                                                                                | Декларативные языки программирования<br>(лаб.)<br>Дубинин В.Н. ауд. 7а-320 |                                                                            |                                                                                          |
|     | 15:35-17:10 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 17:25-19:00 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 19:15-20:50 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
| ВТ  | 8:00-9:35   | Проектирование устройств на программируемых логических интегральных схемах<br>(лек.)<br>Гурин Е.И. ауд. 7б-204 |                                                                            | Декларативные языки программирования<br>(лаб.)<br>Акифьев И.В. ауд. 7а-320 |                                                                                          |
|     | 9:50-11:25  |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 11:40-13:15 |                                                                                                                |                                                                            |                                                                            | Декларативные языки программирования<br>(лаб.)<br>Дубинин В.Н., Акифьев И.В. ауд. 7а-320 |
|     | 13:45-15:20 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 15:35-17:10 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 17:25-19:00 |                                                                                                                |                                                                            |                                                                            |                                                                                          |
|     | 19:15-20:50 |                                                                                                                |                                                                            |                                                                            |                                                                                          |

Рисунок 39 – Студенческая ведомость

Тест 8 – Общие функции.

Валидация конфликтов (автоматическая при любом создании/редактировании):

- занятость аудитории в тот же день, пару.
- занятость группы (нельзя поставить группе два занятия в одно время).

## **4 Охрана труда**

При работе с web-приложением составления расписания основным техническим средством является персональный компьютер (ПК). В процессе эксплуатации на пользователя действуют несколько вредных факторов, связанных с длительным пребыванием в неподвижной позе, напряжением зрения, а также с электромагнитным излучением и возможным перегревом оборудования. Для снижения их влияния необходимо соблюдать ряд организационных и технических мер.

### **4.1 Организация рабочего места**

Стол и стул должны быть подобраны так, чтобы пользователь мог работать без излишнего наклона корпуса. Экран монитора размещается на расстоянии вытянутой руки (около 50-70 см) от глаз. Верхний край экрана - на уровне взгляда или чуть ниже. Клавиатура и мышь располагаются на такой высоте, чтобы предплечья были параллельны полу, а запястья не прогибались. Ноги пользователя должны полностью стоять на полу (или на подставке). Такая эргономика позволяет избежать хронических болей в спине, шее и плечах.

Помещение должно иметь равномерное естественное и искусственное освещение. Прямые солнечные лучи и блики на экране недопустимы - они резко повышают утомляемость глаз. Желательно использовать матовые поверхности стен и рабочего стола, чтобы исключить отражения. В комнате с несколькими компьютерами важно обеспечить нормальный воздухообмен (проветривание или кондиционирование), поскольку техника нагревается и снижает влажность воздуха. Оптимальная температура - около 22-24 °С, влажность – 40-60 %. Сухой воздух приводит к пересыханию слизистых глаз и быстрой усталости.

### **4.2 Режим труда и профилактика утомления**

Наиболее эффективный способ снизить нагрузку - делать короткие перерывы. Каждый час работы желательно прерываться на 5-7 минут. В это время полезно встать, пройтись, размять шею и плечи, сделать несколько простых упражнений для глаз: переводить взгляд с близкого предмета на

дальний, моргать, вращать глазными яблоками. Длительная непрерывная работа за ПК не должна превышать двух часов. Общая продолжительность работы в течение дня - не более шести часов при стандартной восьмичасовой смене (оставшееся время можно использовать для работы с документами, совещаний или других задач без экрана). Такой подход заметно снижает риск синдрома «сухого глаза», головных болей и снижения концентрации.

#### **4.3 Электробезопасность и пожарная безопасность**

Вся компьютерная техника должна подключаться к исправным розеткам с заземляющим контактом. Не допускается использование повреждённых кабелей, переходников и удлинителей. Кабели не должны проходить под ногами или через дверные проёмы - это создаёт риск споткнуться и повредить изоляцию. Запрещается самостоятельно разбирать блоки питания или другие внутренние компоненты ПК без отключения от сети. При появлении дыма, искр или необычного запаха нужно немедленно обесточить оборудование и вызвать специалиста.

Помещение не должно загромождаться горючими материалами (бумага, мебель) вблизи электрообогревателей или вентиляционных отверстий ПК. Проходы к дверям и к огнетушителю всегда должны быть свободны. Для тушения возможного возгорания используется углекислотный огнетушитель (порошковый не рекомендуется из-за повреждения техники). Курение и использование открытого огня в помещении категорически запрещены. Один раз в год рекомендуется проверять сопротивление изоляции электропроводки.

## **Заключение**

В рамках выполненной выпускной квалификационной работы разработано полнофункциональное web-приложение для автоматизированного составления семестрового расписания вуза на основе файла поручений. Программа самостоятельно извлекает из документа все необходимые данные: академические группы, дисциплины, преподавателей, аудитории с их оборудованием. Парсер корректно обрабатывает объединённые ячейки, многострочные записи и диапазоны групп, благодаря чему процесс переноса нагрузки из учебного отдела в систему занимает секунды вместо нескольких дней ручной работы.

Разработанная система предоставляет два типа учётных записей: администратор и преподаватель. Администратор может загружать поручения, управлять справочниками (группы, аудитории, дисциплины), создавать и редактировать пользователей, просматривать расписание любого преподавателя и утверждать либо отклонять готовые варианты с возможностью предложить правки. Преподаватель, в свою очередь, видит только свои назначения и в интерактивной сетке самостоятельно выбирает время и аудиторию для каждого занятия. Программа в реальном времени проверяет корректность выбора: нельзя поставить два занятия в одну аудиторию одновременно, назначить группе больше четырёх пар в день. При возникновении конфликта система выводит понятное сообщение и блокирует сохранение, что полностью исключает ошибки, неизбежные при ручном составлении расписания.

Особенностью приложения является поддержка совместных лабораторных занятий, когда одну группу ведут два преподавателя. В таких случаях лектор создаёт занятие, а приглашённый преподаватель получает уведомление и может либо принять его, либо отклонить, после чего система автоматически разделяет группу на две подгруппы, и каждый преподаватель работает со своей частью студентов. Этот механизм решает реальную проблему, с которой сталкиваются многие кафедры, и отсутствует в большинстве аналогичных продуктов.

Готовое расписание можно экспортировать в Excel в двух форматах: общая преподавательская ведомость, где строки - преподаватели, а столбцы - дни и

пары, и студенческая ведомость с разбивкой по группам и по чётным/нечётным неделям. Оба формата оформлены в соответствии с принятыми в вузе стандартами и готовы к печати или распространению.

Главное преимущество разработанной программы - децентрализация процесса. В отличие от традиционной модели, где диспетчер вручную составляет расписание, а преподаватели потом инициируют множество переносов, в данной системе преподаватели сами планируют свою нагрузку в рамках заданных ограничений. Администратор лишь контролирует и утверждает итоговый вариант. Это снижает нагрузку на учебный отдел в разы, повышает удовлетворённость преподавателей и делает расписание более гибким и реалистичным.

В перспективе приложение можно развивать в нескольких направлениях. Во-первых, расширить парсер для работы с другими форматами файлов, например Excel или CSV, если учебный отдел перейдёт на них. Во-вторых, интегрировать систему с электронным деканатом для автоматического получения актуальных списков групп и студентов. В-третьих, добавить модуль уведомлений: преподаватели и студенты смогут получать сообщения об изменениях в расписании по электронной почте или через push-уведомления. Также возможна разработка мобильного приложения для студентов, где они могли бы смотреть актуальное расписание своей группы, а для преподавателей - быстро просматривать свои занятия и вносить небольшие корректировки. Наконец, можно внедрить аналитику загруженности аудиторий и преподавателей, чтобы оптимизировать распределение нагрузки на следующий семестр.

## Список использованных источников

1. Флэнаган Д. JavaScript. Полное руководство. 7-е изд. – СПб: Питер, 2021 – 732 с.
2. Алексей В. Изучаем программирование на Python. – Эксмо: Питер, 2025 – 616 с.
3. Официальная документация FastAPI. [Электронный ресурс] / Режим доступа: <https://fastapi.tiangolo.com/learn>, свободный (дата обращения 10.04.2026).
4. Официальная документация PostgreSQL. [Электронный ресурс] / Режим доступа: <https://www.postgresql.org/docs/current/index.html>, свободный (дата обращения 15.04.2026).
5. Официальная документация React. [Электронный ресурс] / Режим доступа: <https://react.dev/learn>, свободный (дата обращения 16.04.2026).
6. Официальная документация Bootstrap 5. [Электронный ресурс] / Режим доступа: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>, свободный (дата обращения 17.04.2026).
7. Официальная документация Swagger. [Электронный ресурс] / Режим доступа: <https://swagger.io/docs>, свободный (дата обращения 18.04.2026).

## **Приложение А**

### **(обязательное)**

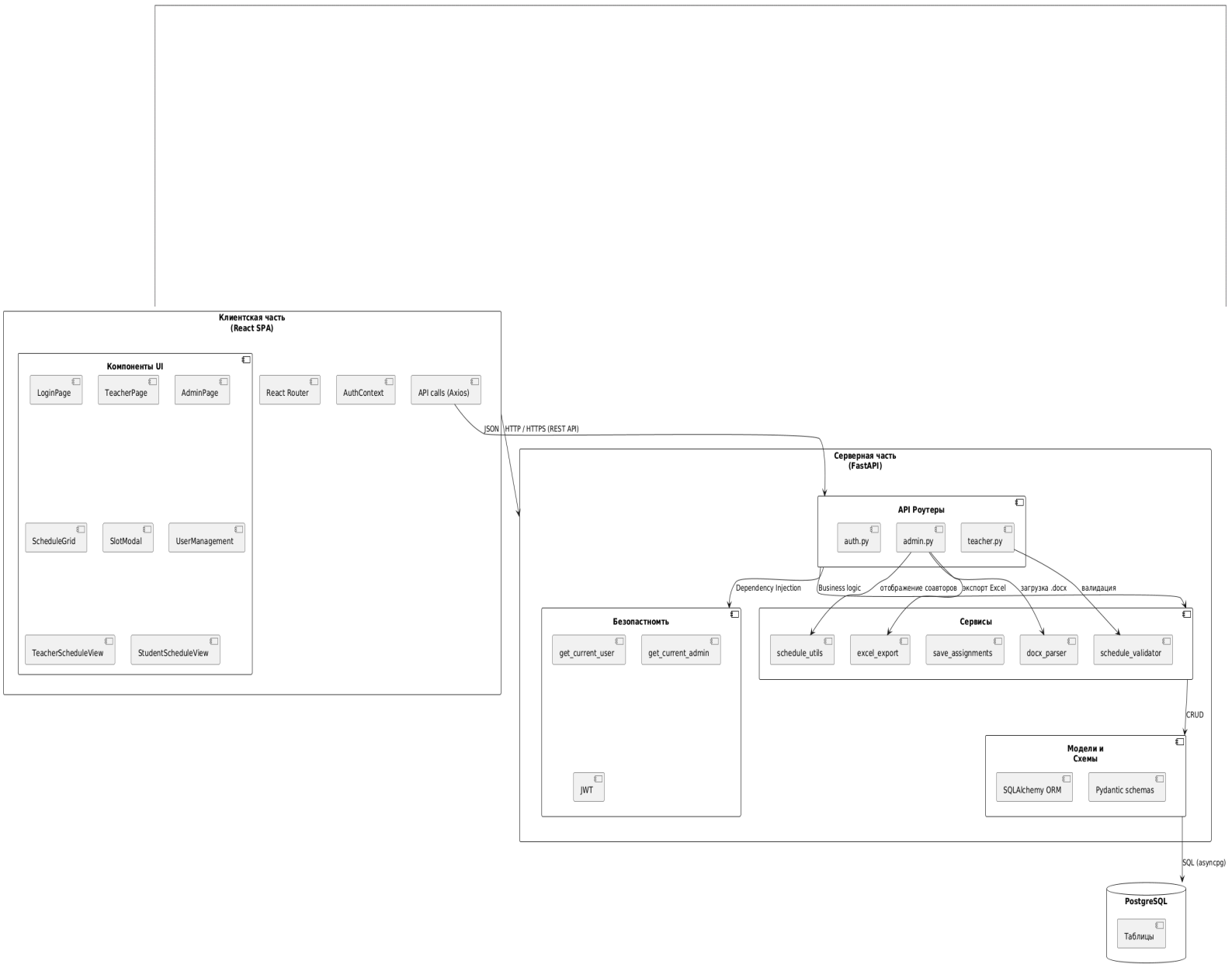
Web-приложение составления расписания ВУЗа

UML-диаграммы

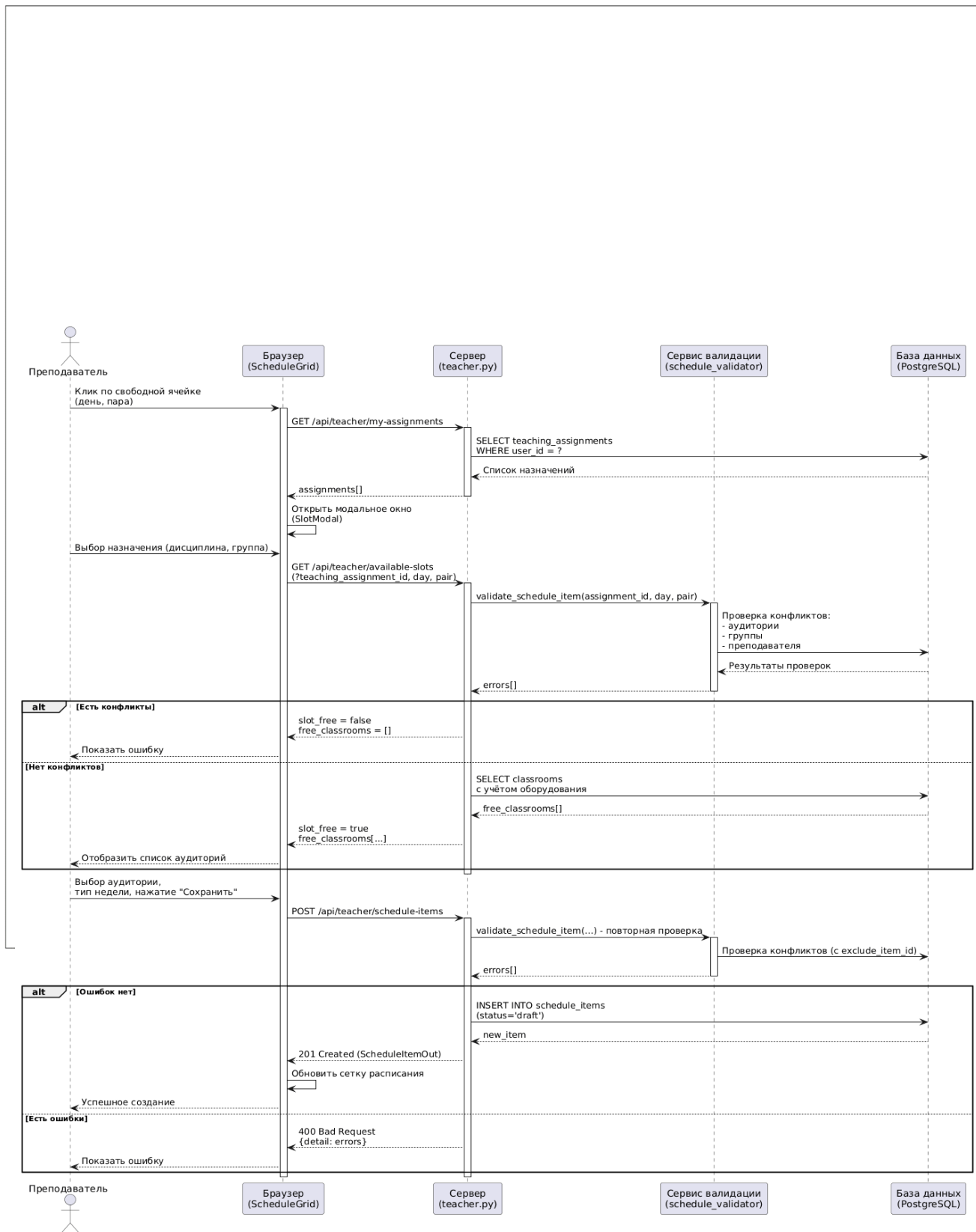
## Приложение А.1 – UML-диаграмма прецедентов использования



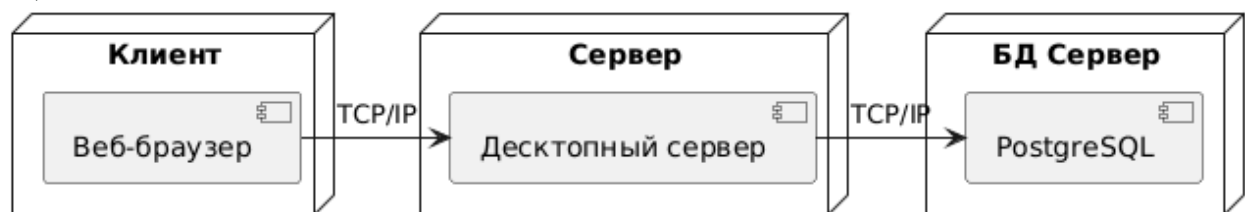
Приложение A.2 – UML-диаграмма классов



## Приложение А.3 – UML-диаграмма последовательности



## Приложение А.4 – UML-диаграмма развертывания



## **Приложение Б**

**(обязательное)**

Web-приложение составления расписания ВУЗа

Листинг программы

## Файл «main.py»

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from app.core.database import engine, Base
from app.api.auth import router as auth_router
from app.api.admin import router as admin_router
from app.api.teacher import router as teacher_router

app = FastAPI(title="University Schedule API")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.include_router(auth_router)
app.include_router(admin_router)
app.include_router(teacher_router)

@app.on_event("startup")
async def startup():
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)

@app.get("/api/health")
async def health():
    return {"status": "ok"}
```

## Файл «schedule\_validator.py»

```
from typing import List, Optional
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select, func
from app.models.schedule_item import ScheduleItem
from app.models.teaching_assignment import TeachingAssignment
from app.models.classroom import Classroom

async def validate_schedule_item(
    db: AsyncSession,
    teaching_assignment_id: int,
    classroom_id: int,
    day_of_week: int,
    pair_number: int,
    exclude_item_id: Optional[int] = None,
    teacher_id_override: Optional[int] = None,
    group_id_override: Optional[int] = None,
    lesson_type_override: Optional[str] = None,
```

```

        collaboration_group_id: Optional[int] = None,
    ) -> List[str]:
        errors = []

        assignment = await db.get(TeachingAssignment,
teaching_assignment_id)
        if not assignment:
            errors.append("Назначение не найдено")
            return errors

        teacher_id = teacher_id_override or assignment.user_id
        group_id = group_id_override or assignment.group_id
        lesson_type = lesson_type_override or assignment.lesson_type

        # 1. Занятость аудитории
        classroom_conflict = select(ScheduleItem).where(
            ScheduleItem.classroom_id == classroom_id,
            ScheduleItem.day_of_week == day_of_week,
            ScheduleItem.pair_number == pair_number,
            ScheduleItem.status.in_(["pending", "approved"])
        )
        if exclude_item_id:
            classroom_conflict =
classroom_conflict.where(ScheduleItem.id != exclude_item_id)
        if (await db.execute(classroom_conflict.limit(1))).scalar():
            errors.append("Аудитория уже занята в это время")

        # 2. Занятость группы (с исключением collaboration-группы)
        group_conflict = (
            select(ScheduleItem)
            .join(TeachingAssignment)
            .where(
                TeachingAssignment.group_id == group_id,
                ScheduleItem.day_of_week == day_of_week,
                ScheduleItem.pair_number == pair_number,
                ScheduleItem.status.in_(["pending", "approved", "draft",
"collaboration_invited"])
            )
        )
        if exclude_item_id:
            group_conflict = group_conflict.where(ScheduleItem.id !=
exclude_item_id)
        if collaboration_group_id is not None:
            group_conflict = group_conflict.where(
                ScheduleItem.collaboration_master_id !=
collaboration_group_id,
                ScheduleItem.id != collaboration_group_id
            )
        if (await db.execute(group_conflict.limit(1))).scalar():
            errors.append("Группа уже имеет занятие в это время")

        # 3. Занятость преподавателя (с исключением collaboration-
группы)

```

```

teacher_conflict = (
    select(ScheduleItem)
    .join(TeachingAssignment)
    .where(
        TeachingAssignment.user_id == teacher_id,
        ScheduleItem.day_of_week == day_of_week,
        ScheduleItem.pair_number == pair_number,
        ScheduleItem.status.in_(["pending", "approved", "draft",
"collaboration_invited"])
    )
)
if exclude_item_id:
    teacher_conflict = teacher_conflict.where(ScheduleItem.id !=
exclude_item_id)
if collaboration_group_id is not None:
    teacher_conflict = teacher_conflict.where(
        ScheduleItem.collaboration_master_id !=
collaboration_group_id,
        ScheduleItem.id != collaboration_group_id
    )
if (await db.execute(teacher_conflict.limit(1))).scalar():
    errors.append("Преподаватель уже занят в это время")

# 4. Дневной лимит группы
count_q = (
    select(func.count())
    .select_from(ScheduleItem)
    .join(TeachingAssignment)
    .where(
        TeachingAssignment.group_id == group_id,
        ScheduleItem.day_of_week == day_of_week,
        ScheduleItem.status.in_(["pending", "approved", "draft",
"collaboration_invited"])
    )
)
if exclude_item_id:
    count_q = count_q.where(ScheduleItem.id != exclude_item_id)
if collaboration_group_id is not None:
    count_q = count_q.where(
        ScheduleItem.collaboration_master_id !=
collaboration_group_id,
        ScheduleItem.id != collaboration_group_id
    )
current_count = (await db.execute(count_q)).scalar()
if current_count >= 4:
    errors.append(f"У группы не может быть больше 4 пар в день
(сейчас {current_count})")

# 5. Оборудование аудитории
classroom = await db.get(Classroom, classroom_id)
if not classroom:
    errors.append("Аудитория не найдена")
return errors

```

```

        if lesson_type == "lecture" and not classroom.has_projector:
            errors.append("Для лекции требуется проектор")
        if lesson_type in ("lab", "practice") and not
classroom.has_computers:
            errors.append("Для лабораторной работы нужны компьютеры")

    return errors

```

Файл «schedule\_utils.py»

```

from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select
from app.models.schedule_item import ScheduleItem
from app.models.user import User
from typing import List, Dict

async def get_co_teachers_for_master(db: AsyncSession, master_id:
int) -> List[str]:
    """
    Возвращает список ФИО преподавателей, которые приняли
приглашение
    для данного мастер-занятия (collaboration_master_id ==
master_id) .
    """
    stmt = select(ScheduleItem).where(
        ScheduleItem.collaboration_master_id == master_id,
        ScheduleItem.invited_user_id.isnot(None)
    )
    invites = await db.execute(stmt)
    invites = invites.scalars().all()

    co_names = []
    for inv in invites:
        if inv.invited_user_id:
            user = await db.get(User, inv.invited_user_id)
            if user:
                co_names.append(user.full_name)
    return co_names

async def get_teacher_display_name(db: AsyncSession, item:
ScheduleItem) -> str:
    """
    Возвращает строку с ФИО лектора и всех принявших приглашение
соавторов.
    """
    ta = item.teaching_assignment
    if not ta or not ta.teacher:
        return ""

    main_name = ta.teacher.full_name

```

```

co_names = []

if item.collaboration_master_id is None:
    # Мастер-запись (создана лектором)
    co_names = await get_co_teachers_for_master(db, item.id)
elif item.collaboration_master_id is not None and
item.invited_user_id is not None:
    # Приглашение - показываем всех, включая лектора из мастер-
записи
    master = await db.get(ScheduleItem,
item.collaboration_master_id)
    if master and master.teaching_assignment and
master.teaching_assignment.teacher:
        main_name = master.teaching_assignment.teacher.full_name
        co_names = await get_co_teachers_for_master(db,
master.id)
        # Добавляем текущего приглашённого, если он не учтён
        user = await db.get(User, item.invited_user_id)
        if user and user.full_name not in co_names:
            co_names.append(user.full_name)

    all_names = [main_name] + [n for n in co_names if n !=
main_name]
    return ", ".join(all_names)

async def enrich_items_with_co_teachers(db: AsyncSession, items:
List[ScheduleItem]) -> Dict[int, str]:
    """
    Для списка ScheduleItem возвращает словарь {item.id:
teacher_names_str}.
    Оптимизирует количество запросов к БД.
    """
    result = {}
    master_ids = [item.id for item in items if
item.collaboration_master_id is None]
    if not master_ids:
        for item in items:
            result[item.id] = await get_teacher_display_name(db,
item)
    return result

# Загружаем все приглашения для этих мастеров (без фильтра по
статусу)
stmt = select(ScheduleItem).where(
    ScheduleItem.collaboration_master_id.in_(master_ids),
    ScheduleItem.invited_user_id.isnot(None)
)
invites = await db.execute(stmt)
invites = invites.scalars().all()

# Группируем преподавателей по master_id
invites_by_master: Dict[int, List[User]] = {}

```

```

    for inv in invites:
        if inv.invited_user_id:
            user = await db.get(User, inv.invited_user_id)
            if user:

invites_by_master.setdefault(inv.collaboration_master_id,
[]) .append(user)

    for item in items:
        if item.collaboration_master_id is None:
            ta = item.teaching_assignment
            main_name = ta.teacher.full_name if ta and ta.teacher
        else:
            co_names = [u.full_name for u in
invites_by_master.get(item.id, [])]
            all_names = [main_name] + [n for n in co_names if n !=
main_name]
            result[item.id] = ", ".join(all_names)
        else:
            result[item.id] = await get_teacher_display_name(db,
item)
    return result

```

Файл «save\_assignments.py»

```

from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.future import select
from app.models.student_group import StudentGroup
from app.models.classroom import Classroom
from app.models.discipline import Discipline
from app.models.user import User
from app.models.teaching_assignment import TeachingAssignment
from app.core.security import hash_password
from typing import Dict, Any

async def save_parsed_data(data: Dict[str, Any], db: AsyncSession)
-> Dict[str, int]:
    stats = {'groups': 0, 'classrooms': 0, 'disciplines': 0,
'teachers': 0, 'assignments': 0}

    # Группы
    for gname in data.get('groups', []):
        res = await
db.execute(select(StudentGroup).where(StudentGroup.name == gname))
        if not res.scalar_one_or_none():
            db.add(StudentGroup(name=gname))
            stats['groups'] += 1

    # Аудитории
    for room in data.get('classrooms', []):

```

```

        res = await
db.execute(select(Classroom).where(Classroom.number ==
room['number']))
    existing = res.scalar_one_or_none()
    if not existing:
        db.add(Classroom(
            number=room['number'],
            has_projector=room.get('has_projector', False),
            has_computers=room.get('has_computers', False)
        ))
        stats['classrooms'] += 1
    else:
        existing.has_projector = room.get('has_projector',
existing.has_projector)
        existing.has_computers = room.get('has_computers',
existing.has_computers)

# Дисциплины
for disc in data.get('disciplines', []):
    res = await
db.execute(select(Discipline).where(Discipline.name ==
disc['name']))
    if not res.scalar_one_or_none():
        db.add(Discipline(
            name=disc['name'],
            semester=disc.get('semester', ''),
            hours_lecture=disc.get('hours_lecture', 0),
            hours_lab=disc.get('hours_lab', 0),
            hours_practice=disc.get('hours_practice', 0)
        ))
        stats['disciplines'] += 1

# Преподаватели
for tdata in data.get('teachers', []):
    full_name = tdata['full_name']
    result = await db.execute(select(User).where(User.full_name
== full_name))
    user = result.scalar_one_or_none()
    if not user:
        username = full_name.lower().replace(' ', '_')
        user = User(
            username=username,
            password_hash=hash_password('teacher123'),
            full_name=full_name,
            role='teacher',
            academic_degree=tdata.get('degree', ''),
            department=tdata.get('department', '')
        )
        db.add(user)
        stats['teachers'] += 1
    else:
        user.academic_degree = tdata.get('degree', '') or
user.academic_degree

```

```

        user.department = tdata.get('department', '') or
user.department

    await db.flush()

    # Назначения
    for assign in data.get('assignments', []):
        user_res = await
db.execute(select(User).where(User.full_name ==
assign['teacher_name']))
        user = user_res.scalar_one_or_none()
        if not user:
            continue
        disc_res = await
db.execute(select(Discipline).where(Discipline.name ==
assign['discipline_name']))
        disc = disc_res.scalar_one_or_none()
        if not disc:
            continue
        group_res = await
db.execute(select(StudentGroup).where(StudentGroup.name ==
assign['group_name']))
        group = group_res.scalar_one_or_none()
        if not group:
            continue

    # Получаем ID дополнительных преподавателей
    secondary_ids = []
    if assign.get('secondary_teacher_names'):
        for s_name in assign['secondary_teacher_names']:
            s_user = await
db.execute(select(User).where(User.full_name == s_name))
            s_user = s_user.scalar_one_or_none()
            if s_user:
                secondary_ids.append(s_user.id)

    exist = await db.execute(
        select(TeachingAssignment).where(
            TeachingAssignment.user_id == user.id,
            TeachingAssignment.discipline_id == disc.id,
            TeachingAssignment.group_id == group.id,
            TeachingAssignment.lesson_type ==
assign['lesson_type']
        )
    )
    if exist.scalar_one_or_none():
        continue

    db.add(TeachingAssignment(
        user_id=user.id,
        discipline_id=disc.id,
        group_id=group.id,
        lesson_type=assign['lesson_type'],

```

```

        total_hours=assign.get('total_hours', 0),
        secondary_teacher_ids=secondary_ids
    ))
    stats['assignments'] += 1

    await db.commit()
    return stats

```

### Файл «excel\_export.py»

```

import openpyxl
from openpyxl.styles import Font, Alignment, Border, Side,
PatternFill
from openpyxl.utils import get_column_letter
from io import BytesIO
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select
from sqlalchemy.orm import joinedload
from app.models.schedule_item import ScheduleItem
from app.models.teaching_assignment import TeachingAssignment
from app.services.schedule_utils import
enrich_items_with_co_teachers

# ----- Стили -----
thin_side = Side(style='thin')
thick_side = Side(style='medium')
thin_border = Border(left=thin_side, right=thin_side, top=thin_side,
bottom=thin_side)
day_sep_border = Border(left=thin_side, right=thick_side,
top=thin_side, bottom=thin_side)

header_fill = PatternFill(start_color="D9D9D9", end_color="D9D9D9",
fill_type="solid")
header_font = Font(name='Calibri', bold=True, size=10)
normal_font = Font(name='Calibri', size=9)
small_font = Font(name='Calibri', size=8)
wrap_alignment = Alignment(wrap_text=True, vertical='top',
horizontal='left')
center_alignment = Alignment(horizontal='center', vertical='center')
time_center_alignment = Alignment(horizontal='center',
vertical='center', wrap_text=True)

def auto_fit_row_heights(ws, max_chars_per_line=30):
    """
    Автоподбор высоты строк: определяет максимальное количество
    строк текста
    в ячейках строки (с учётом объединения) и устанавливает высоту.
    """
    for row in ws.iter_rows():
        max_lines = 1

```

```

        for cell in row:
            if isinstance(cell, openpyxl.cell.cell.MergedCell):
                continue
            if cell.value and isinstance(cell.value, str):
                lines = cell.value.split('\n')
                # Учитываем возможный перенос по ширине столбца
                total_lines = 0
                for line in lines:
                    if len(line) > max_chars_per_line:
                        total_lines += (len(line) //
max_chars_per_line) + 1
                    else:
                        total_lines += 1
                max_lines = max(max_lines, total_lines)
            if max_lines > 1:
                ws.row_dimensions[row[0].row].height = max(15, max_lines
* 15)

```

```

async def generate_schedule_excel(db: AsyncSession, user_id: int |
None = None):
    """Общая ведомость преподавателей с пометкой недели и полными
    границами."""
    base_query = (
        select(ScheduleItem)
        .options(
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.discipline),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.group),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.teacher),
            joinedload(ScheduleItem.classroom),
        )
        .where(ScheduleItem.status == 'approved')
    )

    if user_id is not None:
        base_query =
base_query.join(TeachingAssignment).where(TeachingAssignment.user_id
== user_id)

    result = await db.execute(base_query)
    items = result.unique().scalars().all()

    teacher_name_map = await enrich_items_with_co_teachers(db,
items)
    teacher_schedule = {}
    for item in items:
        teacher = item.teaching_assignment.teacher
        if not teacher:
            continue
        t_id = teacher.id

```

```

        if t_id not in teacher_schedule:
            teacher_schedule[t_id] = {'teacher': teacher, 'items':
[]}

            teacher_schedule[t_id]['items'].append(item)

wb = openpyxl.Workbook()
ws = wb.active
ws.title = "Расписание"

days = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница',
'суббота']
slots_per_day = 7
total_cols = 2 + len(days) * slots_per_day # 44

# Заголовок
ws.merge_cells(start_row=1, start_column=1, end_row=1,
end_column=total_cols)
main_title = "Расписание занятий на кафедре \"Вычислительная
техника\" на весенний семестр 2025-2026 учебного года"
ws.cell(row=1, column=1, value=main_title).font =
Font(name='Calibri', bold=True, size=14)
ws['A1'].alignment = center_alignment
ws.row_dimensions[1].height = 35

# Пустая строка 2
ws.append([])

# Строка 3: дни недели (объединённые)
ws.merge_cells('A3:A4')
ws.merge_cells('B3:B4')
ws['A3'] = '№ п/п'
ws['B3'] = 'Ф.И.О.'
ws['A3'].font = header_font
ws['B3'].font = header_font
ws['A3'].alignment = center_alignment
ws['B3'].alignment = center_alignment
ws['A3'].fill = header_fill
ws['B3'].fill = header_fill
ws['A3'].border = thin_border
ws['B3'].border = thin_border

col = 3
for day in days:
    ws.merge_cells(start_row=3, start_column=col, end_row=3,
end_column=col + slots_per_day - 1)
    cell = ws.cell(row=3, column=col, value=day.capitalize())
    cell.font = header_font
    cell.alignment = center_alignment
    cell.fill = header_fill
    for c in range(col, col + slots_per_day):
        border = day_sep_border if c == col + slots_per_day - 1
else thin_border
        ws.cell(row=3, column=c).border = border

```

```

        col += slots_per_day

# Строка 4: время
time_slots = ['8.00-9.35', '9.50-11.25', '11.40-13.15',
               '13.45-15.20', '15.35-17.10', '17.25-19.00',
               '19.15-20.50']
col = 3
for day_idx in range(len(days)):
    for s_idx, slot in enumerate(time_slots):
        cell = ws.cell(row=4, column=col, value=slot)
        cell.font = Font(name='Calibri', size=8)
        cell.alignment = center_alignment
        cell.fill = header_fill
        border = day_sep_border if s_idx == len(time_slots) - 1
    else thin_border
        cell.border = border
        col += 1

# Данные преподавателей
row_num = 5
for idx, (t_id, t_data) in enumerate(teacher_schedule.items(),
start=1):
    teacher = t_data['teacher']
    ws.cell(row=row_num, column=1, value=idx).font = normal_font
    ws.cell(row=row_num, column=1).alignment = center_alignment
    ws.cell(row=row_num, column=1).border = thin_border
    ws.cell(row=row_num, column=2, value=teacher.full_name).font
= normal_font
    ws.cell(row=row_num, column=2).alignment =
Alignment(vertical='top', horizontal='left')
    ws.cell(row=row_num, column=2).border = thin_border

    schedule_matrix = {day: {p: '' for p in
range(slots_per_day)} for day in days}
    for item in t_data['items']:
        day_idx = item.day_of_week - 1
        if 0 <= day_idx < len(days):
            day_name = days[day_idx]
            pair_idx = item.pair_number - 1
            if 0 <= pair_idx < slots_per_day:
                week_note = ''
                if item.week_type == 'odd':
                    week_note = ' (неч.)'
                elif item.week_type == 'even':
                    week_note = ' (чет.)'
                teacher_names = teacher_name_map.get(item.id,
teacher.full_name)
                info = (f"{item.teaching_assignment.group.name
if item.teaching_assignment.group else ''} "

f"{item.teaching_assignment.discipline.name if
item.teaching_assignment.discipline else ''} "

```

```

            f"{'лк.' if
item.teaching_assignment.lesson_type == 'lecture' else 'лб.'} "
            f"{teacher_names} "
            f"айд.{item.classroom.number if
item.classroom else ''}"
            f"{week_note}")
        schedule_matrix[day_name][pair_idx] = info

        col = 3
        for day in days:
            for pair in range(slots_per_day):
                cell = ws.cell(row=row_num, column=col,
value=schedule_matrix[day][pair])
                cell.font = Font(name='Calibri', size=8)
                cell.alignment = wrap_alignment
                border = day_sep_border if pair == slots_per_day - 1
else thin_border
                cell.border = border
                col += 1
            row_num += 1

        # Автоподбор высоты
        auto_fit_row_heights(ws)

        # Ширина столбцов
        ws.column_dimensions['A'].width = 6
        ws.column_dimensions['B'].width = 22
        for c in range(3, total_cols + 1):
            ws.column_dimensions[get_column_letter(c)].width = 13

        ws.sheet_view.showGridLines = False

        output = BytesIO()
        wb.save(output)
        output.seek(0)
        return output

```

```

async def generate_student_schedule_excel(db: AsyncSession,
group_id: int | None = None):

```

```

    """Студенческое расписание с правильным разделением по
неделям."""

```

```

    base_query = (
        select(ScheduleItem)
        .options(
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.discipline),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.group),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.teacher),
            joinedload(ScheduleItem.classroom),
        )
    )

```

```

        .where(ScheduleItem.status == 'approved')
    )
    if group_id is not None:
        base_query =
base_query.join(TeachingAssignment).where(TeachingAssignment.group_id == group_id)

    result = await db.execute(base_query)
    items = result.unique().scalars().all()

    # Получаем имена преподавателей (соавторов)
    from app.services.schedule_utils import
enrich_items_with_co_teachers
    teacher_name_map = await enrich_items_with_co_teachers(db,
items)

    schedule_by_group = {}
    for item in items:
        group = item.teaching_assignment.group
        if not group:
            continue
        g_id = group.id
        if g_id not in schedule_by_group:
            schedule_by_group[g_id] = {'group': group, 'items': []}
        schedule_by_group[g_id]['items'].append(item)

    days_short = ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ']
    time_slots = [
        '8:00-9:35', '9:50-11:25', '11:40-13:15',
        '13:45-15:20', '15:35-17:10', '17:25-19:00', '19:15-20:50'
    ]

    groups = sorted(schedule_by_group.values(), key=lambda x:
x['group'].name)
    if not groups:
        wb = openpyxl.Workbook()
        ws = wb.active
        ws.title = "Расписание"
        ws['A1'] = "Нет утверждённых занятий"
        output = BytesIO()
        wb.save(output)
        output.seek(0)
        return output

    num_groups = len(groups)
    cols_per_group = 3
    total_data_cols = num_groups * cols_per_group

    wb = openpyxl.Workbook()
    ws = wb.active
    ws.title = "Расписание"

    # Гриф утверждения (как в оригинале)

```

```

    ws.merge_cells(start_row=1, start_column=1, end_row=1,
end_column=total_data_cols)
    ws['A1'] = '"Утверждаю" Ректор ПГУ'
    ws['A1'].font = Font(name='Calibri', bold=True, size=14)
    ws['A1'].alignment = center_alignment

    ws.merge_cells(start_row=2, start_column=1, end_row=2,
end_column=total_data_cols)
    ws['A2'] = '/А.Д.Гуляков/'
    ws['A2'].font = Font(name='Calibri', bold=True, size=12)
    ws['A2'].alignment = center_alignment

    ws.merge_cells(start_row=3, start_column=1, end_row=3,
end_column=total_data_cols)
    ws['A3'] = 'Начальник УМУ /В.А.Соловьев/'
    ws['A3'].font = Font(name='Calibri', size=11)
    ws['A3'].alignment = center_alignment

    ws.merge_cells(start_row=4, start_column=1, end_row=4,
end_column=total_data_cols)
    ws['A4'] = '26.01.2026'
    ws['A4'].font = Font(name='Calibri', size=10)
    ws['A4'].alignment = center_alignment

    ws.append([]) # строка 5

    ws.merge_cells(start_row=6, start_column=1, end_row=6,
end_column=total_data_cols)
    title = "Расписание занятий ФВТ на весенний семестр 2025-2026
уч.г."
    ws.cell(row=6, column=1, value=title).font =
Font(name='Calibri', bold=True, size=14)
    ws['A6'].alignment = center_alignment
    ws.append([]) # строка 7

    # Заголовки групп
    row_num = 8
    header_row = []
    for g in groups:
        header_row.extend(['дни', 'часы', g['group'].name])
    for col_idx, value in enumerate(header_row, start=1):
        cell = ws.cell(row=row_num, column=col_idx, value=value)
        cell.font = header_font
        cell.alignment = center_alignment
        cell.fill = header_fill
        cell.border = thin_border
    row_num += 1

    # Данные
    for day_idx, day in enumerate(days_short):
        day_num = day_idx + 1
        day_start_row = row_num

```

```

        for pair_idx, slot_time in enumerate(time_slots):
            pair_num = pair_idx + 1
            upper_data = [] # список кортежей: (odd_text,
even_text, all_flag)
            for g in groups:
                item_odd = next((it for it in g['items']
                                if it.day_of_week == day_num
                                and it.pair_number == pair_num
                                and it.week_type == 'odd'), None)
                item_even = next((it for it in g['items']
                                if it.day_of_week == day_num
                                and it.pair_number == pair_num
                                and it.week_type == 'even'), None)
                item_all = next((it for it in g['items']
                                if it.day_of_week == day_num
                                and it.pair_number == pair_num
                                and it.week_type == 'all'), None)

                if item_all:
                    disc =
f"{item_all.teaching_assignment.discipline.name if
item_all.teaching_assignment.discipline else ''}\n({'лек.' if
item_all.teaching_assignment.lesson_type == 'lecture' else 'лаб.' if
item_all.teaching_assignment.lesson_type == 'lab' else ''})"
                    teacher = f"{teacher_name_map.get(item_all.id,
'')} ауд.{item_all.classroom.number if item_all.classroom else ''}"
                    upper_data.append((None, None,
f"{disc}\n{teacher}"))
                else:
                    odd_text = ''
                    even_text = ''
                    if item_odd:
                        disc_odd =
f"{item_odd.teaching_assignment.discipline.name if
item_odd.teaching_assignment.discipline else ''}\n({'лек.' if
item_odd.teaching_assignment.lesson_type == 'lecture' else 'лаб.' if
item_odd.teaching_assignment.lesson_type == 'lab' else ''})"
                        t_odd = f"{teacher_name_map.get(item_odd.id,
'')} ауд.{item_odd.classroom.number if item_odd.classroom else ''}"
                        odd_text = f"{disc_odd}\n{t_odd}"
                    if item_even:
                        disc_even =
f"{item_even.teaching_assignment.discipline.name if
item_even.teaching_assignment.discipline else ''}\n({'лек.' if
item_even.teaching_assignment.lesson_type == 'lecture' else 'лаб.'
if item_even.teaching_assignment.lesson_type == 'lab' else ''})"
                        t_even =
f"{teacher_name_map.get(item_even.id, '')} ауд.
{item_even.classroom.number if item_even.classroom else ''}"
                        even_text = f"{disc_even}\n{t_even}"
                    upper_data.append((odd_text, even_text, None))

```

# Строим верхнюю и нижнюю строки

```

        upper_row = []
        lower_row = []
        for g_idx, (odd, even, all_text) in
enumerate(upper_data):
            if all_text is not None:
                upper_row.extend([day, slot_time, all_text])
                lower_row.extend(['', '', ''])
            else:
                upper_row.extend([day, slot_time, odd if odd
else ''])
                lower_row.extend(['', '', even if even else ''])

# Записываем верхнюю
for col_idx, value in enumerate(upper_row, start=1):
    cell = ws.cell(row=row_num, column=col_idx,
value=value)
    cell.font = normal_font
    cell.alignment = wrap_alignment
    cell.border = thin_border
    row_num += 1

# Записываем нижнюю
for col_idx, value in enumerate(lower_row, start=1):
    cell = ws.cell(row=row_num, column=col_idx,
value=value)
    cell.font = small_font
    cell.alignment = wrap_alignment
    cell.border = thin_border
    row_num += 1

# Объединяем день и время по вертикали
for g_idx in range(num_groups):
    base_col = g_idx * 3 + 1
    ws.merge_cells(start_row=row_num-2,
start_column=base_col,
end_row=row_num-1,
end_column=base_col)
    ws.merge_cells(start_row=row_num-2,
start_column=base_col+1,
end_row=row_num-1,
end_column=base_col+1)
    ws.cell(row=row_num-2, column=base_col+1).alignment
= time_center_alignment

# Объединяем ячейки all
for g_idx, (_, _, all_text) in enumerate(upper_data):
    if all_text is not None:
        data_col = g_idx * 3 + 3
        ws.merge_cells(start_row=row_num-2,
start_column=data_col,
end_row=row_num-1,
end_column=data_col)

```

```

        # Объединяем день по вертикали внутри дня
        for g_idx in range(num_groups):
            col_day = g_idx * 3 + 1
            if row_num - 1 > day_start_row:
                ws.merge_cells(start_row=day_start_row,
start_column=col_day,
                                end_row=row_num-1,
end_column=col_day)

        # Нижний заголовок
        footer_row = header_row
        for col_idx, value in enumerate(footer_row, start=1):
            cell = ws.cell(row=row_num, column=col_idx, value=value)
            cell.font = header_font
            cell.alignment = center_alignment
            cell.fill = header_fill
            cell.border = thin_border
        row_num += 1

        # Автоподбор высоты
        auto_fit_row_heights(ws)

        # Ширина столбцов
        for c in range(1, total_data_cols + 1):
            letter = get_column_letter(c)
            if c % 3 == 1:
                ws.column_dimensions[letter].width = 4
            elif c % 3 == 2:
                ws.column_dimensions[letter].width = 10
            else:
                ws.column_dimensions[letter].width = 18

        ws.sheet_view.showGridLines = False

        output = BytesIO()
        wb.save(output)
        output.seek(0)
        return output

```

Файл «docx\_parser.py»

```

import re
from docx import Document
from typing import List, Dict, Any, Optional

def _normalize_teacher_name(raw_name: str) -> str:
    name = raw_name.strip().rstrip(',')
    # Удаляем учёные степени и звания
    name = re.sub(r'\s+(ст\.|с*)?препод(аватель)?\.\?', '', name)
    name = re.sub(r'\s+(к\.|с*[тэп]\.\s*н\.\s*,?\s*)?(доцент|
профессор|ассистент|ст\.\s*препод\.\?)\.\?', '', name)

```

```

    name = re.sub(r',?\s*(к\.\w+\.\s*н\.\s*,?\s*)*(доцент|профессор|
ассистент|ст\.\s*препод\.)?', '', name)
    name = re.sub(r'\s+к\.п\.\n\.', '', name)
    name = re.sub(r'\s+к\.\э\.\n\.', '', name)
    # Приводим к формату "Фамилия И.О."
    match = re.match(r'([А-ЯЁ][а-яё]+\s*([А-ЯЁ]\.)\s*([А-ЯЁ]\.)?)',
name)
    if match:
        last = match.group(1)
        first = match.group(2) if match.group(2) else ''
        middle = match.group(3) if match.group(3) else ''
        return f"{last} {first}{middle}".strip()
    return ' '.join(name.split())

def _normalize_room_number(room: str) -> str:
    """Приводит номер аудитории к единому формату: русская 'а',
дефис, цифры."""
    room = room.strip()
    room = room.replace('а', 'a').replace('А', 'A')
    room = room.replace(' ', '')
    room = re.sub(r'([0-9]+)([А-Яа-я])', lambda m: m.group(1) +
m.group(2).lower(), room)
    return room

def _expand_group_range(group_str: str) -> List[str]:
    group_str = group_str.strip().rstrip(',')
    if '-' not in group_str:
        return [group_str]
    match = re.match(r'^(.+?)(\d+)-(\d+)$', group_str)
    if not match:
        return [group_str]
    base = match.group(1)
    start = int(match.group(2))
    end = int(match.group(3))
    return [f"{base}{i}" for i in range(start, end + 1)]

def _extract_groups_from_text(text: str) -> List[str]:
    groups = []
    group_pattern = re.compile(r'\b(\d{2}[А-ЯЁА-Z]+[\d]+
(?:-\d+)?)\b')
    matches = group_pattern.findall(text)
    for match in matches:
        groups.extend(_expand_group_range(match))
    return groups

def _extract_hours(text: str) -> int:
    match = re.search(r'\b(\d+)\b', text)
    return int(match.group(1)) if match else 0

def _extract_name(text: str) -> Optional[str]:
    match = re.search(r'([А-ЯЁ][а-яё]+\s[А-ЯЁ]\.[А-ЯЁ]\.?)', text)
    if match:
        return _normalize_teacher_name(match.group(1))

```

```

return None

def parse_assignments(file_path: str) -> Dict[str, Any]:
    doc = Document(file_path)
    if not doc.tables:
        raise ValueError("В документе нет таблиц")

    disciplines_list = []
    groups_set = set()
    teachers_dict: Dict[str, Dict] = {}
    assignments = []
    room_set = set() # просто номера аудиторий

    # Сбор строк из ВСЕХ таблиц
    all_rows = []
    for table_idx, table in enumerate(doc.tables):
        for row in table.rows:
            cells = row.cells
            if len(cells) < 6:
                continue
            row_text = [c.text.strip() for c in cells[:6]]
            first = row_text[0]
            if first.startswith('+') or first.startswith('|') or
all(cell == ' ' for cell in row_text):
                continue
            if first in ('1-5 курсы бак, спец.', 'Магистранты',
'Заочно', 'ПОРУЧЕНИЯ', 'преподавателям кафедры'):
                continue
            all_rows.append(row_text)

    print(f"[DEBUG] Собрано строк из всех таблиц: {len(all_rows)}")

    def is_new_discipline(row):
        disc = row[0]
        if not disc:
            return False
        if re.match(r'^\d{2}[А-ЯЁА-Z]+[\d]+$', disc):
            return False
        if disc.isdigit():
            return False
        lower = disc.lower()
        if lower in ('лекции', 'лабораторные', 'курсы', 'поручения',
'преподавателям', 'магистранты', 'заочно'):
            return False
        return True

    current_discipline = None
    current_groups = []
    current_lecture_hours = 0
    current_lecturer_name = None
    current_notes = ""
    current_rooms = ""

```

```

def flush_current():
    nonlocal current_discipline, current_groups,
current_lecture_hours, current_lecturer_name, current_notes,
current_rooms
    if not current_discipline or not current_groups:
        return
    # Лекции
    if current_lecturer_name and current_lecture_hours > 0:
        for grp in current_groups:
            assignments.append({
                'teacher_name': current_lecturer_name,
                'discipline_name': current_discipline,
                'group_name': grp,
                'lesson_type': 'lecture',
                'total_hours': current_lecture_hours,
                'notes': current_notes,
                'rooms': current_rooms,
            })
        if not any(d['name'] == current_discipline for d in
disciplines_list):
            disciplines_list.append({
                'name': current_discipline,
                'semester': '2 семестр 2025/2026',
                'hours_lecture': current_lecture_hours,
                'hours_lab': 0,
                'hours_practice': 0,
            })
        groups_set.update(current_groups)

for idx, row in enumerate(all_rows):
    disc, grps, lect, lab, notes, rooms = row
    # Собираем аудитории из этой строки (только номера, без
флагов)
    if rooms:
        for rm in re.findall(r'\d[a-яёА-ЯЁа-яА-Я]?-\d+', rooms):
            room_set.add(_normalize_room_number(rm))

    if is_new_discipline(row):
        flush_current()
        current_discipline = disc
        current_lecture_hours = _extract_hours(lect)
        current_lecturer_name = _extract_name(lect)
        current_notes = notes
        current_rooms = rooms
        current_groups = _extract_groups_from_text(grps) if grps
    else []
        if current_lecturer_name and current_lecturer_name not
in teachers_dict:
            teachers_dict[current_lecturer_name] = {
                'full_name': current_lecturer_name,
                'degree': '',
                'department': 'Кафедра ВТ'
            }

```

```

else:
    if grps:

current_groups.extend(_extract_groups_from_text(grps))
    if notes and not current_notes:
        current_notes = notes
    if rooms and not current_rooms:
        current_rooms = rooms

# Лабораторные
if lab and current_discipline:
    lab_hours = _extract_hours(lab)
    for d in disciplines_list:
        if d['name'] == current_discipline:
            d['hours_lab'] = max(d.get('hours_lab', 0),
lab_hours)
            break
    parts = re.split(r'(\b\d{2}[A-ЯЁА-Z]+[\d]+(?:-\d+)?\b)',
lab)

    current_group_lab = None
    for part in parts:
        part = part.strip()
        if not part:
            continue
        if re.match(r'\b\d{2}[A-ЯЁА-Z]+[\d]+(?:-\d+)?\b',
part):
            current_group_lab = part
        else:
            names = re.findall(r'([A-ЯЁ][a-яё]+\s[A-ЯЁ]\.[A-
ЯЁ]\.?)', part)

            if not names:
                continue
            main_teacher = _normalize_teacher_name(names[0])
            secondary = [_normalize_teacher_name(n) for n in
names[1:]]

            for nm in names:
                normalized = _normalize_teacher_name(nm)
                if normalized not in teachers_dict:
                    teachers_dict[normalized] = {
                        'full_name': normalized,
                        'degree': '',
                        'department': 'Кафедра ВТ'
                    }

            groups_to_assign =
_expand_group_range(current_group_lab) if current_group_lab else
current_groups

            for grp in groups_to_assign:
                assignments.append({
                    'teacher_name': main_teacher,
                    'secondary_teacher_names': secondary,
                    'discipline_name': current_discipline,
                    'group_name': grp,
                    'lesson_type': 'lab',

```

```

        'total_hours': lab_hours,
        'notes': notes,
        'rooms': rooms,
    })
    groups_set.add(grp)

flush_current()

# ----- Парсинг примечаний для аудиторий -----
notes_full = []
start_collecting = False
for paragraph in doc.paragraphs:
    text = paragraph.text.strip()
    if 'Примечания' in text or 'Аудитории:' in text:
        start_collecting = True
    if start_collecting:
        notes_full.append(text)
full_notes = '\n'.join(notes_full)

# Инициализируем свойства аудиторий
room_props = {room: {'has_projector': False, 'has_computers':
False} for room in room_set}

# 1) Разбираем основное описание аудиторий (после "Аудитории:")
main_desc_match = re.search(r'Аудитории:(.*)$', full_notes,
re.MULTILINE)
if main_desc_match:
    main_desc = main_desc_match.group(1)
    # Разбиваем на части по запятым и "и"
    statements = re.split(r',\s*|\s+и\s+', main_desc)
    for stmt in statements:
        stmt = stmt.strip()
        rooms_in_stmt = re.findall(r'(\d[a-яёА-ЯЁа-зА-З]?-\d+)',
stmt)

        if not rooms_in_stmt:
            continue
        # Определяем флаги для этих аудиторий
        has_proj = 'есть проектор' in stmt or 'проектор' in stmt
        has_comp = 'компьютер' in stmt
        if 'нет проектора' in stmt:
            has_proj = False
        if 'только компьютеры' in stmt:
            has_comp = True
            has_proj = False
        for rm in rooms_in_stmt:
            norm_rm = _normalize_room_number(rm)
            if norm_rm in room_props:
                room_props[norm_rm]['has_projector'] = has_proj
                room_props[norm_rm]['has_computers'] = has_comp

# 2) Аудитории для лабораторных работ (добавляем компьютеры)
lab_match = re.search(r'Аудитории для лабораторных работ\s*(с
компьютерами)\s*(.*)', full_notes)

```

```

    if lab_match:
        lab_list = lab_match.group(1)
        lab_rooms = re.findall(r'(\d[a-яёА-ЯЁа-зА-З]?-\d+)',
lab_list)
        for rm in lab_rooms:
            norm_rm = _normalize_room_number(rm)
            if norm_rm in room_props:
                room_props[norm_rm]['has_computers'] = True

for room in ['76-204', '76-202']:
    norm_rm = _normalize_room_number(room)
    if norm_rm in room_props:
        room_props[norm_rm]['has_projector'] = True
        # компьютеры оставляем как есть (обычно false)

# Формируем итоговый список аудиторий
classrooms_list = []
for room, props in room_props.items():
    classrooms_list.append({
        'number': room,
        'has_projector': props['has_projector'],
        'has_computers': props['has_computers']
    })

print(f"[DEBUG] Итог: групп={len(groups_set)},
назначений={len(assignments)}, дисциплин={len(disciplines_list)},
аудиторий={len(classrooms_list)}")
return {
    'assignments': assignments,
    'groups': sorted(groups_set),
    'teachers': sorted(teachers_dict.values(), key=lambda x:
x['full_name']),
    'disciplines': disciplines_list,
    'classrooms': classrooms_list,
}

```

Файл «user\_schemas.py»

```

from pydantic import BaseModel
from typing import Optional

class UserCreate(BaseModel):
    username: str
    password: str
    full_name: str

class UserResponse(BaseModel):
    id: int
    username: str
    full_name: str
    role: str

```

```

class Config:
    from_attributes = True

class Token(BaseModel):
    access_token: str
    token_type: str

```

Файл «teacher\_schemas.py»

```

from pydantic import BaseModel
from typing import Optional
from datetime import datetime

class TeacherAssignmentOut(BaseModel):
    id: int
    discipline_name: str
    group_name: str
    lesson_type: str
    total_hours: int

    class Config:
        from_attributes = True

    @staticmethod
    def from_orm(obj):
        return TeacherAssignmentOut(
            id=obj.id,
            discipline_name=obj.discipline.name if obj.discipline
else "",
            group_name=obj.group.name if obj.group else "",
            lesson_type=obj.lesson_type,
            total_hours=obj.total_hours,
        )

class ClassroomOut(BaseModel):
    id: int
    number: str
    has_projector: bool
    has_computers: bool
    suitable: Optional[bool] = True

    class Config:
        from_attributes = True

class ScheduleItemOut(BaseModel):
    id: int
    teaching_assignment_id: int
    classroom_id: int
    day_of_week: int
    pair_number: int
    week_type: str

```

```

status: str
created_at: Optional[datetime] = None
discipline_name: Optional[str] = None
group_name: Optional[str] = None
lesson_type: Optional[str] = None
teacher_name: Optional[str] = None
classroom_number: Optional[str] = None
submit_note: Optional[str] = None
rejection_reason: Optional[str] = None
proposed_by_admin: Optional[str] = None
invited_user_id: Optional[int] = None
collaboration_master_id: Optional[int] = None
collaboration_history: Optional[list] = None

class Config:
    from_attributes = True

class ScheduleItemCreate(BaseModel):
    teaching_assignment_id: int
    classroom_id: int
    day_of_week: int
    pair_number: int
    week_type: Optional[str] = "all"

class AvailableSlotsResponse(BaseModel):
    slot_free: bool
    free_classrooms: list[ClassroomOut]

```

Файл «admin\_schemas.py»

```

from pydantic import BaseModel
from typing import Optional

# --- Group ---
class GroupCreate(BaseModel):
    name: str
    course: int
    specialty: str

class GroupUpdate(BaseModel):
    name: Optional[str] = None
    course: Optional[int] = None
    specialty: Optional[str] = None

class GroupOut(BaseModel):
    id: int
    name: str
    course: Optional[int] = None
    specialty: Optional[str] = None
    class Config:
        from_attributes = True

```

```

# --- Classroom ---
class ClassroomCreate(BaseModel):
    number: str
    capacity: int = 0
    has_projector: bool = False
    has_computers: bool = False
    building: Optional[str] = None

class ClassroomUpdate(BaseModel):
    number: Optional[str] = None
    capacity: Optional[int] = None
    has_projector: Optional[bool] = None
    has_computers: Optional[bool] = None
    building: Optional[str] = None

class ClassroomOut(BaseModel):
    id: int
    number: str
    capacity: int
    has_projector: bool
    has_computers: bool
    building: Optional[str] = None
    class Config:
        from_attributes = True

# --- Discipline ---
class DisciplineCreate(BaseModel):
    name: str
    semester: Optional[str] = None
    hours_lecture: int = 0
    hours_lab: int = 0
    hours_practice: int = 0

class DisciplineUpdate(BaseModel):
    name: Optional[str] = None
    semester: Optional[str] = None
    hours_lecture: Optional[int] = None
    hours_lab: Optional[int] = None
    hours_practice: Optional[int] = None

class DisciplineOut(BaseModel):
    id: int
    name: str
    semester: Optional[str] = None
    hours_lecture: int
    hours_lab: int
    hours_practice: int
    class Config:
        from_attributes = True

# --- Teacher (User with role=teacher) ---
class TeacherOut(BaseModel):
    id: int

```

```

    username: str
    full_name: str
    role: str
    academic_degree: Optional[str] = None
    department: Optional[str] = None
    class Config:
        from_attributes = True

class TeacherCreate(BaseModel):
    username: str
    password: str
    full_name: str
    academic_degree: Optional[str] = None
    department: Optional[str] = None

# --- User management ---
class UserOut(BaseModel):
    id: int
    username: str
    full_name: str
    role: str
    academic_degree: Optional[str] = None
    department: Optional[str] = None
    class Config:
        from_attributes = True

class UserCreate(BaseModel):
    username: str
    password: str
    full_name: str
    role: str = 'teacher'
    academic_degree: Optional[str] = None
    department: Optional[str] = None

class UserUpdate(BaseModel):
    username: Optional[str] = None
    full_name: Optional[str] = None
    role: Optional[str] = None
    academic_degree: Optional[str] = None
    department: Optional[str] = None
    password: Optional[str] = None    # новый пароль, если нужно
    сменить

```

Файл «user.py»

```

from sqlalchemy import Column, Integer, String, Enum, JSON
import enum
from app.core.database import Base

class UserRole(str, enum.Enum):
    teacher = "teacher"
    admin = "admin"

```

```

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(50), unique=True, nullable=False,
index=True)
    password_hash = Column(String(255), nullable=False)
    full_name = Column(String(100), nullable=False)
    role = Column(Enum(UserRole), nullable=False,
default=UserRole.teacher)

    # Поля преподавателя (могут быть NULL для админов)
    academic_degree = Column(String(50))
    department = Column(String(100))
    constraints_json = Column(JSON, default=dict)

```

Файл «teaching\_assignment.py»

```

from sqlalchemy import Column, Integer, String, ForeignKey,
UniqueConstraint, JSON
from sqlalchemy.orm import relationship
from app.core.database import Base

class TeachingAssignment(Base):
    __tablename__ = "teaching_assignments"

    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey("users.id"),
nullable=False)
    discipline_id = Column(Integer, ForeignKey("disciplines.id"),
nullable=False)
    group_id = Column(Integer, ForeignKey("student_groups.id"),
nullable=False)
    lesson_type = Column(String(20), nullable=False)
    total_hours = Column(Integer, default=0)

    secondary_teacher_ids = Column(JSON, default=list)

    teacher = relationship("User", backref="assignments")
    discipline = relationship("Discipline", backref="assignments")
    group = relationship("StudentGroup", backref="assignments")

    __table_args__ = (
        UniqueConstraint('user_id', 'discipline_id', 'group_id',
'lesson_type', name='uq_teaching_assignment'),
    )

```

Файл «student\_group.py»

```

from sqlalchemy import Column, Integer, String
from app.core.database import Base

```

```

class StudentGroup(Base):
    __tablename__ = "student_groups"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(20), unique=True, nullable=False,
index=True)
    course = Column(Integer)
    specialty = Column(String(100))

```

Файл «schedule\_item.py»

```

from sqlalchemy import Column, Integer, String, ForeignKey,
DateTime, Text, CheckConstraint, JSON
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from app.core.database import Base

```

```

class ScheduleItem(Base):
    __tablename__ = "schedule_items"

    id = Column(Integer, primary_key=True, index=True)
    teaching_assignment_id = Column(Integer,
ForeignKey("teaching_assignments.id"), nullable=False)
    classroom_id = Column(Integer, ForeignKey("classrooms.id"),
nullable=False)
    day_of_week = Column(Integer, nullable=False)
    pair_number = Column(Integer, nullable=False)
    week_type = Column(String(10), default="all")
    status = Column(String(30), default="draft")
    created_by = Column(Integer, ForeignKey("users.id"))
    created_at = Column(DateTime(timezone=True),
server_default=func.now())
    updated_at = Column(DateTime(timezone=True),
onupdate=func.now())

    submit_note = Column(Text, nullable=True)
    rejection_reason = Column(Text, nullable=True)
    proposed_by_admin = Column(String(30), nullable=True)

    invited_user_id = Column(Integer, ForeignKey("users.id"),
nullable=True)
    collaboration_master_id = Column(Integer,
ForeignKey("schedule_items.id"), nullable=True)
    # История сообщений: список {author_id, author_name, message,
timestamp}
    collaboration_history = Column(JSON, default=list)

    teaching_assignment = relationship("TeachingAssignment")
    classroom = relationship("Classroom")

    __table_args__ = (

```

```

        CheckConstraint('pair_number >= 1 AND pair_number <= 6',
name='check_pair_number'),
    )

```

Файл «discipline.py»

```

from sqlalchemy import Column, Integer, String
from app.core.database import Base

class Discipline(Base):
    __tablename__ = "disciplines"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(200), nullable=False)
    semester = Column(String(20))
    hours_lecture = Column(Integer, default=0)
    hours_lab = Column(Integer, default=0)
    hours_practice = Column(Integer, default=0)

```

Файл «classroom.py»

```

from sqlalchemy import Column, Integer, String, Boolean
from app.core.database import Base

class Classroom(Base):
    __tablename__ = "classrooms"

    id = Column(Integer, primary_key=True, index=True)
    number = Column(String(20), unique=True, nullable=False)
    capacity = Column(Integer, default=0)
    has_projector = Column(Boolean, default=False)
    has_computers = Column(Boolean, default=False)
    building = Column(String(20))

```

Файл «security.py»

```

from datetime import datetime, timedelta
from jose import jwt
from passlib.context import CryptContext
from app.core.config import settings

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def hash_password(password: str) -> str:
    return pwd_context.hash(password)

def verify_password(plain_password: str, hashed_password: str) -> bool:
    return pwd_context.verify(plain_password, hashed_password)

```

```

def create_access_token(data: dict) -> str:
    to_encode = data.copy()
    if "sub" in to_encode and not isinstance(to_encode["sub"], str):
        to_encode["sub"] = str(to_encode["sub"])
    expire = datetime.utcnow() +
timedelta(minutes=settings.ACCESS_TOKEN_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    return jwt.encode(to_encode, settings.SECRET_KEY,
algorithm="HS256")

def create_refresh_token(data: dict) -> str:
    to_encode = data.copy()
    if "sub" in to_encode and not isinstance(to_encode["sub"], str):
        to_encode["sub"] = str(to_encode["sub"])
    expire = datetime.utcnow() +
timedelta(days=settings.REFRESH_TOKEN_EXPIRE_DAYS)
    to_encode.update({"exp": expire})
    return jwt.encode(to_encode, settings.SECRET_KEY,
algorithm="HS256")

```

Файл «database.py»

```

from sqlalchemy.ext.asyncio import create_async_engine,
AsyncSession, async_sessionmaker
from sqlalchemy.orm import DeclarativeBase
from app.core.config import settings

engine = create_async_engine(settings.DATABASE_URL, echo=True)
async_session = async_sessionmaker(engine, class_=AsyncSession,
expire_on_commit=False)

class Base(DeclarativeBase):
    pass

async def get_db() -> AsyncSession:
    async with async_session() as session:
        yield session

```

Файл «config.py»

```

from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    DATABASE_URL: str
    SECRET_KEY: str
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 30
    REFRESH_TOKEN_EXPIRE_DAYS: int = 7

    class Config:
        env_file = ".env"
        env_file_encoding = "utf-8"

```

```
settings = Settings()
```

Файл «teacher.py»

```
from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.future import select
from sqlalchemy.orm import joinedload
from fastapi.responses import StreamingResponse
from typing import List, Optional
from datetime import datetime
import app.schemas.teacher_schemas as schemas
from app.core.database import get_db
from app.api.dependencies import get_current_user
from app.models.user import User
from app.models.teaching_assignment import TeachingAssignment
from app.models.schedule_item import ScheduleItem
from app.models.classroom import Classroom
from app.models.student_group import StudentGroup
from app.services.schedule_validator import validate_schedule_item
from app.services.schedule_utils import get_teacher_display_name,
enrich_items_with_co_teachers

router = APIRouter(prefix="/api/teacher", tags=["teacher"])

def _add_history(current_history: list, author: User, action: str,
note: Optional[str] = None):
    history = current_history or []
    history.append({
        "author_id": author.id,
        "author_name": author.full_name,
        "action": action,
        "note": note or "",
        "timestamp": datetime.utcnow().isoformat()
    })
    return history

@router.get("/my-assignments",
response_model=List[schemas.TeacherAssignmentOut])
async def my_assignments(
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    direct = await db.execute(
        select(TeachingAssignment)
        .options(joinedload(TeachingAssignment.discipline),
joinedload(TeachingAssignment.group))
        .where(TeachingAssignment.user_id == current_user.id)
    )
```

```

        direct = direct.unique().scalars().all()

        all_with_secondary = await db.execute(
            select(TeachingAssignment)
                .options(joinedload(TeachingAssignment.discipline),
joinedload(TeachingAssignment.group))
                .where(TeachingAssignment.secondary_teacher_ids.isnot(None))
        )
        all_with_secondary = all_with_secondary.unique().scalars().all()
        additional = [ta for ta in all_with_secondary if current_user.id
in (ta.secondary_teacher_ids or [])]

        all_assignments = list(direct) + additional
        result = []
        for ta in all_assignments:
            co_teachers = []
            for tid in (ta.secondary_teacher_ids or []):
                if tid != current_user.id:
                    t = await db.get(User, tid)
                    if t:
                        co_teachers.append(t.full_name)
            if ta.user_id != current_user.id:
                main_teacher = await db.get(User, ta.user_id)
                co_teachers.append(main_teacher.full_name if
main_teacher else 'Неизвестно')
            result.append({
                'id': ta.id,
                'discipline_name': ta.discipline.name if ta.discipline
else '',
                'group_name': ta.group.name if ta.group else '',
                'lesson_type': ta.lesson_type,
                'total_hours': ta.total_hours,
                'co_teachers': co_teachers
            })
        return result

@router.get("/my-schedule",
response_model=List[schemas.ScheduleItemOut])
async def my_schedule(
    week_type: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    owner_query = (
        select(ScheduleItem)
            .join(TeachingAssignment)
            .where(
                TeachingAssignment.user_id == current_user.id,
                ScheduleItem.invited_user_id == None
            )
            .options(
                joinedload(ScheduleItem.teaching_assignment)

```

```

        .joinedload(TeachingAssignment.discipline),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.group),
        joinedload(ScheduleItem.classroom),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.teacher)
    )
)
if week_type in ("odd", "even"):
    owner_query =
owner_query.where(ScheduleItem.week_type.in_([week_type, "all"]))

invited_query = (
    select(ScheduleItem)
    .join(TeachingAssignment,
ScheduleItem.teaching_assignment_id == TeachingAssignment.id)
    .options(
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.discipline),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.group),
        joinedload(ScheduleItem.classroom),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.teacher)
    )
    .where(
        ScheduleItem.invited_user_id == current_user.id,
        ScheduleItem.created_by != current_user.id
    )
)
if week_type in ("odd", "even"):
    invited_query =
invited_query.where(ScheduleItem.week_type.in_([week_type, "all"]))

owner_items = (await
db.execute(owner_query)).unique().scalars().all()
invited_items = (await
db.execute(invited_query)).unique().scalars().all()
all_items = list(owner_items) + list(invited_items)

# Получаем словарь с именами преподавателей для каждого элемента
teacher_name_map = await enrich_items_with_co_teachers(db,
all_items)

result = []
for item in all_items:
    ta = item.teaching_assignment
    result.append({
        "id": item.id,
        "teaching_assignment_id": item.teaching_assignment_id,
        "classroom_id": item.classroom_id,
        "day_of_week": item.day_of_week,
        "pair_number": item.pair_number,

```

```

        "week_type": item.week_type,
        "status": item.status,
        "created_at": item.created_at,
        "discipline_name": ta.discipline.name if ta.discipline
    else "",
        "group_name": ta.group.name if ta.group else "",
        "lesson_type": ta.lesson_type,
        "teacher_name": teacher_name_map.get(item.id, ""),
        "classroom_number": item.classroom.number if
item.classroom else "",
        "submit_note": item.submit_note,
        "rejection_reason": item.rejection_reason,
        "proposed_by_admin": item.proposed_by_admin,
        "invited_user_id": item.invited_user_id,
        "collaboration_master_id": item.collaboration_master_id,
        "collaboration_history": item.collaboration_history or
    [],
    })
    return result

```

```

@router.get("/available-slots",
response_model=schemas.AvailableSlotsResponse)
async def available_slots(
    teaching_assignment_id: int = Query(...),
    day_of_week: int = Query(...),
    pair_number: int = Query(...),
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    assignment = await db.get(TeachingAssignment,
teaching_assignment_id)
    if not assignment:
        raise HTTPException(status_code=404, detail="Назначение не
найдено")
    if assignment.user_id != current_user.id and current_user.id not
in (assignment.secondary_teacher_ids or []):
        raise HTTPException(status_code=403, detail="Нет доступа")

    errors = await validate_schedule_item(
        db, teaching_assignment_id, classroom_id=0,
        day_of_week=day_of_week, pair_number=pair_number
    )
    slot_free = not any(e for e in errors if "аудитория" not in
e.lower())

    lesson_type = assignment.lesson_type
    all_rooms_q = select(Classroom)
    all_rooms = (await db.execute(all_rooms_q)).scalars().all()

    free_classrooms = []
    for room in all_rooms:
        conflict = await db.execute(

```

```

        select(ScheduleItem).where(
            ScheduleItem.classroom_id == room.id,
            ScheduleItem.day_of_week == day_of_week,
            ScheduleItem.pair_number == pair_number,
            ScheduleItem.status.in_(["pending", "approved"])
        )
    )
    if not conflict.scalar_one_or_none():
        suitable = True
        if lesson_type == "lecture" and not room.has_projector:
            suitable = False
        elif lesson_type in ("lab", "practice") and not
room.has_computers:
            suitable = False
        free_classrooms.append({
            "id": room.id,
            "number": room.number,
            "has_projector": room.has_projector,
            "has_computers": room.has_computers,
            "suitable": suitable,
        })

    return schemas.AvailableSlotsResponse(
        slot_free=slot_free,
        free_classrooms=free_classrooms
    )

@router.post("/schedule-items",
response_model=schemas.ScheduleItemOut)
async def create_schedule_item(
    item: schemas.ScheduleItemCreate,
    note: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    assignment = await db.get(TeachingAssignment,
item.teaching_assignment_id)
    if not assignment:
        raise HTTPException(status_code=404, detail="Назначение не
найдено")

    is_collaborative = False
    other_teachers_ids = []

    if assignment.lesson_type == "lab" and
assignment.secondary_teacher_ids:
        lecturer_assignment = await db.execute(
            select(TeachingAssignment).where(
                TeachingAssignment.discipline_id ==
assignment.discipline_id,
                TeachingAssignment.group_id == assignment.group_id,
                TeachingAssignment.lesson_type == "lecture",

```

```

        TeachingAssignment.user_id == current_user.id
    )
)
if not lecturer_assignment.scalar_one_or_none():
    raise HTTPException(status_code=403, detail="Только
лектор может создавать совместные лабораторные занятия")
is_collaborative = True
other_teachers_ids = assignment.secondary_teacher_ids
else:
    if assignment.user_id != current_user.id and current_user.id
not in (assignment.secondary_teacher_ids or []):
        raise HTTPException(status_code=403, detail="Нет доступа
к этому назначению")

if is_collaborative:
    busy_teachers = []
    for s_tid in other_teachers_ids:
        conflict = await validate_schedule_item(
            db, item.teaching_assignment_id, item.classroom_id,
            item.day_of_week, item.pair_number,
            teacher_id_override=s_tid
        )
        if any("Преподаватель" in e for e in conflict):
            teacher = await db.get(User, s_tid)
            busy_teachers.append(teacher.full_name if teacher
else f"ID {s_tid}")
        if busy_teachers:
            raise HTTPException(
                status_code=400,
                detail=f"Преподаватели заняты в это время: {'',
'.join(busy_teachers)}"
            )

    errors = await validate_schedule_item(
        db, item.teaching_assignment_id, item.classroom_id,
        item.day_of_week, item.pair_number
    )
    if errors:
        raise HTTPException(status_code=400, detail=";
".join(errors))

    status = "collaboration_waiting" if is_collaborative else
"draft"

schedule_item = ScheduleItem(
    teaching_assignment_id=item.teaching_assignment_id,
    classroom_id=item.classroom_id,
    day_of_week=item.day_of_week,
    pair_number=item.pair_number,
    week_type=item.week_type or "all",
    status=status,
    created_by=current_user.id,
    submit_note=note,

```

```

        collaboration_history=[
            {
                "author_id": current_user.id,
                "author_name": current_user.full_name,
                "action": "created",
                "note": note or "",
                "timestamp": datetime.utcnow().isoformat()
            }
        ] if is_collaborative else []
    )
    db.add(schedule_item)
    await db.flush()

    if is_collaborative:
        for s_tid in other_teachers_ids:
            invitation = ScheduleItem(
                teaching_assignment_id=item.teaching_assignment_id,
                classroom_id=item.classroom_id,
                day_of_week=item.day_of_week,
                pair_number=item.pair_number,
                week_type=item.week_type or "all",
                status="draft",
                created_by=current_user.id,
                invited_user_id=s_tid,
                collaboration_master_id=schedule_item.id
            )
            db.add(invitation)

    await db.commit()
    await db.refresh(schedule_item)
    return schedule_item


@router.put("/schedule-items/{item_id}",
response_model=schemas.ScheduleItemOut)
async def update_schedule_item(
    item_id: int,
    item_update: schemas.ScheduleItemCreate,
    note: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    db_item = await db.get(ScheduleItem, item_id)
    if not db_item:
        raise HTTPException(status_code=404, detail="Занятие не
        найдено")

    can_edit = False
    if db_item.created_by == current_user.id and db_item.status in
    ("collaboration_waiting", "draft"):
        can_edit = True
    elif db_item.invited_user_id == current_user.id and
    db_item.status in ("draft", "collaboration_waiting"):

```

```

        can_edit = True

    if not can_edit:
        raise HTTPException(status_code=403, detail="Нет доступа к
этому занятию")

    errors = await validate_schedule_item(
        db, item_update.teaching_assignment_id,
item_update.classroom_id,
        item_update.day_of_week, item_update.pair_number,
        exclude_item_id=item_id,
        collaboration_group_id=db_item.collaboration_master_id or
db_item.id
    )
    if errors:
        raise HTTPException(status_code=400, detail=";
".join(errors))

    db_item.teaching_assignment_id =
item_update.teaching_assignment_id
    db_item.classroom_id = item_update.classroom_id
    db_item.day_of_week = item_update.day_of_week
    db_item.pair_number = item_update.pair_number
    db_item.week_type = item_update.week_type or "all"

    if db_item.collaboration_master_id is None:
        if db_item.status == "draft":
            pass
        elif db_item.status == "collaboration_waiting":
            db_item.status = "draft"
            invitations = (await db.execute(

select(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
db_item.id)
                )).scalars().all()
            for inv in invitations:
                inv.status = "draft"
        else:
            if db_item.status == "draft":
                db_item.status = "collaboration_waiting"
                master = await db.get(ScheduleItem,
db_item.collaboration_master_id)
                if master:
                    master.status = "draft"
            elif db_item.status == "collaboration_waiting":
                db_item.status = "draft"
                master = await db.get(ScheduleItem,
db_item.collaboration_master_id)
                if master:
                    master.status = "collaboration_waiting"

    if note:
        db_item.collaboration_history = _add_history(

```

```

        db_item.collaboration_history or [], current_user,
"edited", note
    )
    db_item.submit_note = note

    await db.commit()
    return db_item

@router.delete("/schedule-items/{item_id}")
async def delete_schedule_item(
    item_id: int,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    db_item = await db.get(ScheduleItem, item_id)
    if not db_item:
        raise HTTPException(status_code=404, detail="Занятие не
найджено")

    can_delete = False
    if db_item.created_by == current_user.id and db_item.status in
("draft", "collaboration_waiting"):
        can_delete = True
    elif db_item.invited_user_id == current_user.id:
        can_delete = True

    if not can_delete:
        raise HTTPException(status_code=403, detail="Нет доступа")

    if db_item.collaboration_master_id is None:
        from sqlalchemy import delete
        await db.execute(

delete(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
db_item.id)
        )

        await db.delete(db_item)
        await db.commit()
        return {"message": "Занятие удалено"}

@router.post("/schedule-items/submit")
async def submit_schedule(
    note: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    # Находим все черновики, где пользователь является создателем
или приглашённым
    user_items = await db.execute(
        select(ScheduleItem)

```

```

        .join(TeachingAssignment,
ScheduleItem.teaching_assignment_id == TeachingAssignment.id)
        .where(
            ScheduleItem.status == "draft",
            (
                (TeachingAssignment.user_id == current_user.id) |
                (ScheduleItem.invited_user_id == current_user.id)
            )
        )
    )
    user_items = user_items.scalars().all()

    # Группируем по мастер-записям
    groups = {} # master_id -> список элементов (мастер +
приглашения)
    for item in user_items:
        if item.collaboration_master_id is None:
            master_id = item.id
            if master_id not in groups:
                groups[master_id] = [item]
        else:
            master_id = item.collaboration_master_id
            if master_id not in groups:
                # Загружаем мастера
                master = await db.get(ScheduleItem, master_id)
                if master:
                    groups[master_id] = [master]
                else:
                    continue
            if item not in groups[master_id]:
                groups[master_id].append(item)

    # Проверяем, что все элементы в группе имеют статус draft
    to_submit = []
    for master_id, items_in_group in groups.items():
        all_draft = all(it.status == "draft" for it in
items_in_group)
        if not all_draft:
            continue # можно выбросить ошибку, но для простоты
пропускаем
        to_submit.extend(items_in_group)

    if not to_submit:
        raise HTTPException(status_code=400, detail="Нет черновиков
для отправки")

    # Обновляем статусы
    for item in to_submit:
        item.status = "pending"
        if note is not None:
            item.submit_note = note
        item.rejection_reason = None
        item.proposed_by_admin = None

```

```
        await db.commit()
        return {"message": f"Отправлено на утверждение: {len(to_submit)}
занятий (включая совместные)"}

```

```
@router.post("/schedule-items/{item_id}/accept-collaboration")
async def accept_collaboration(
    item_id: int,
    note: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    invitation = await db.get(ScheduleItem, item_id)
    if not invitation or invitation.status != "draft" or not
invitation.invited_user_id:
        raise HTTPException(status_code=404, detail="Приглашение не
найдено или не может быть принято")

    if invitation.invited_user_id != current_user.id:
        raise HTTPException(status_code=403, detail="Это приглашение
не для вас")

    if invitation.collaboration_master_id:
        master = await db.get(ScheduleItem,
invitation.collaboration_master_id)
        if master:
            master.status = "draft"
            related = await db.execute(

select(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
master.id)

            )
            for rel in related.scalars().all():
                rel.status = "draft"

    invitation.status = "draft"
    if note:
        invitation.collaboration_history = _add_history(
            invitation.collaboration_history or [], current_user,
"accepted", note
        )
        invitation.submit_note = note

    await db.commit()
    return {"message": "Приглашение принято, занятие согласовано"}

```

```
@router.post("/schedule-items/{item_id}/reject-collaboration")
async def reject_collaboration(
    item_id: int,
    reason: Optional[str] = None,
    note: Optional[str] = None,

```

```

db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_user)
):
    invitation = await db.get(ScheduleItem, item_id)
    if not invitation or invitation.status != "draft" or not
invitation.invited_user_id:
        raise HTTPException(status_code=404, detail="Приглашение не
найджено или не может быть отклонено")
        if invitation.invited_user_id != current_user.id:
            raise HTTPException(status_code=403, detail="Это приглашение
не для вас")

    master = await db.get(ScheduleItem,
invitation.collaboration_master_id)
    if not master:
        raise HTTPException(status_code=404, detail="Основное
занятие не найдено")

    from sqlalchemy.orm import selectinload
    ta_query =
select(TeachingAssignment).options(selectinload(TeachingAssignment.g
roup)).where(TeachingAssignment.id == master.teaching_assignment_id)
    ta_result = await db.execute(ta_query)
    assignment = ta_result.scalar_one_or_none()
    if not assignment or not assignment.group:
        raise HTTPException(status_code=400, detail="Назначение или
группа не найдены")

    original_group = assignment.group
    subgroup1_name = f"{original_group.name}/1"
    subgroup2_name = f"{original_group.name}/2"

    sub_exists = await db.execute(

select(StudentGroup).where(StudentGroup.name.in_([subgroup1_name,
subgroup2_name]))
    )
    if sub_exists.scalars().first():
        raise HTTPException(status_code=400, detail="Подгруппы уже
существуют")

    sg1 = StudentGroup(name=subgroup1_name,
course=original_group.course, specialty=original_group.specialty)
    sg2 = StudentGroup(name=subgroup2_name,
course=original_group.course, specialty=original_group.specialty)
    db.add(sg1)
    db.add(sg2)
    await db.flush()

    lecturer_id = assignment.user_id
    secondary_id = current_user.id

    db.add(TeachingAssignment(

```

```

        user_id=lecturer_id,
        discipline_id=assignment.discipline_id,
        group_id=sg1.id,
        lesson_type="lab",
        total_hours=assignment.total_hours
    ))
    db.add(TeachingAssignment(
        user_id=secondary_id,
        discipline_id=assignment.discipline_id,
        group_id=sg2.id,
        lesson_type="lab",
        total_hours=assignment.total_hours
    ))

    from sqlalchemy import delete
    await db.execute(
        delete(ScheduleItem).where(
            ScheduleItem.teaching_assignment_id == assignment.id,
            ScheduleItem.status.in_(["draft",
"collaboration_waiting", "collaboration_invited"])
        )
    )

    await db.delete(assignment)
    await db.commit()
    return {"message": f"Группа разделена на {subgroup1_name} и
{subgroup2_name}. Каждый преподаватель теперь работает со своей
подгруппой."}

@router.post("/clear-proposed")
async def clear_proposed(
    db: AsyncSession = Depends(get_db) ,
    current_user: User = Depends(get_current_user)
):
    items = await db.execute(
        select(ScheduleItem)
        .join(TeachingAssignment)
        .where(TeachingAssignment.user_id == current_user.id,
ScheduleItem.proposed_by_admin != None)
    )
    items = items.scalars().all()
    for item in items:
        item.proposed_by_admin = None
    await db.commit()
    return {"message": f"Сброшено предложений: {len(items)}"}

@router.get("/export-my-schedule")
async def export_my_schedule(
    db: AsyncSession = Depends(get_db) ,
    current_user: User = Depends(get_current_user)
):

```

```

        from app.services.excel_export import generate_schedule_excel
        excel_file = await generate_schedule_excel(db,
user_id=current_user.id)
        return StreamingResponse(
            excel_file,
            media_type="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
            headers={"Content-Disposition": f"attachment;
filename=my_schedule_{current_user.full_name}.xlsx"}
        )

@router.get("/public-schedule-data")
async def public_schedule_data(
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):
    base_query = (
        select(ScheduleItem)
        .options(
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.discipline),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.group),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.teacher),
            joinedload(ScheduleItem.classroom),
        )
        .where(ScheduleItem.status == 'approved')
    )

    result = await db.execute(base_query)
    items = result.unique().scalars().all()

    # Получаем имена преподавателей для каждого элемента
    teacher_name_map = await enrich_items_with_co_teachers(db,
items)

    teacher_schedule = {}
    for item in items:
        teacher = item.teaching_assignment.teacher
        if not teacher:
            continue
        t_id = teacher.id
        if t_id not in teacher_schedule:
            teacher_schedule[t_id] = {'teacher': teacher, 'items':
[]}

        teacher_schedule[t_id]['items'].append(item)

    days = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница',
'суббота']
    time_slots = ['8.00-9.35', '9.50-11.25', '11.40-13.15',

```

```

        '13.45-15.20', '15.35-17.10', '17.25-19.00',
        '19.15-20.50']

    rows = []
    for idx, (t_id, t_data) in enumerate(teacher_schedule.items(),
start=1):
        teacher = t_data['teacher']
        row_data = {
            'num': idx,
            'teacher_name': teacher.full_name,
            'cells': {}
        }
        for item in t_data['items']:
            day_idx = item.day_of_week - 1
            if 0 <= day_idx < len(days):
                pair_idx = item.pair_number - 1
                if 0 <= pair_idx < 7:
                    key = f"{day_idx}-{pair_idx}"
                    week_note = ''
                    if item.week_type == 'odd':
                        week_note = ' (неч.)'
                    elif item.week_type == 'even':
                        week_note = ' (чет.)'
                    teacher_names = teacher_name_map.get(item.id,
teacher.full_name)
                    info = (f"{item.teaching_assignment.group.name
if item.teaching_assignment.group else ''} "
f"{item.teaching_assignment.discipline.name if
item.teaching_assignment.discipline else ''} "
                        f"{'лк.' if
item.teaching_assignment.lesson_type == 'lecture' else 'лб.'} "
                        f"{teacher_names} "
                        f"айд.{item.classroom.number if
item.classroom else ''}")
                    f"{week_note}")
                    row_data['cells'][key] = info
                rows.append(row_data)

    return {
        'days': days,
        'time_slots': time_slots,
        'rows': rows
    }

```

Файл «dependencies.py»

```

from fastapi import Depends, HTTPException, status
from fastapi.security import HTTPAuthorizationCredentials,
HTTPBearer
from jose import JWTError, jwt
from sqlalchemy.ext.asyncio import AsyncSession

```

```

from sqlalchemy.future import select
from app.core.database import get_db
from app.core.config import settings
from app.models.user import User

security_scheme = HTTPBearer()

async def get_current_user(
    credentials: HTTPAuthorizationCredentials =
Depends(security_scheme),
    db: AsyncSession = Depends(get_db)
) -> User:
    token = credentials.credentials
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = jwt.decode(token, settings.SECRET_KEY,
algorithms=["HS256"])
        user_id = int(payload.get("sub"))
        if user_id is None:
            raise credentials_exception
    except (JWTError, ValueError):
        raise credentials_exception

    result = await db.execute(select(User).where(User.id ==
user_id))
    user = result.scalar_one_or_none()
    if user is None:
        raise credentials_exception
    return user

async def get_current_admin(current_user: User =
Depends(get_current_user)) -> User:
    if current_user.role != "admin":
        raise HTTPException(status_code=status.HTTP_403_FORBIDDEN,
detail="Not enough permissions")
    return current_user

```

Файл «auth.py»

```

from fastapi import APIRouter, Depends, HTTPException, status
from fastapi.security import OAuth2PasswordRequestForm
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.future import select

from app.core.database import get_db
from app.core.security import hash_password, verify_password,
create_access_token, create_refresh_token
from app.models.user import User, UserRole

```

```

from app.schemas.user_schemas import UserCreate, UserResponse, Token
from app.api.dependencies import get_current_user

router = APIRouter(prefix="/api/auth", tags=["auth"])

@router.post("/register", response_model=UserResponse)
async def register(user_data: UserCreate, db: AsyncSession =
Depends(get_db)):
    existing = await db.execute(select(User).where(User.username ==
user_data.username))
    if existing.scalar_one_or_none():
        raise HTTPException(status_code=400, detail="Username
already registered")

    user = User(
        username=user_data.username,
        password_hash=hash_password(user_data.password),
        full_name=user_data.full_name,
        role=UserRole.teacher    # по умолчанию
    )
    db.add(user)
    await db.commit()
    await db.refresh(user)
    return user

@router.post("/login", response_model=Token)
async def login(form_data: OAuth2PasswordRequestForm = Depends(),
db: AsyncSession = Depends(get_db)):
    result = await db.execute(select(User).where(User.username ==
form_data.username))
    user = result.scalar_one_or_none()
    if not user or not verify_password(form_data.password,
user.password_hash):
        raise HTTPException(status_code=401, detail="Invalid
credentials")

    access_token = create_access_token(data={"sub": str(user.id),
"role": user.role})
    refresh_token = create_refresh_token(data={"sub": str(user.id)})
    return {"access_token": access_token, "token_type": "bearer"}

@router.get("/me", response_model=UserResponse)
async def read_current_user(current_user: User =
Depends(get_current_user)):
    return current_user

@router.put("/change-password")
async def change_password(
    old_password: str,
    new_password: str,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_user)
):

```

```

        if not verify_password(old_password,
current_user.password_hash):
            raise HTTPException(status_code=400, detail="Старый пароль
неверен")
        current_user.password_hash = hash_password(new_password)
        await db.commit()
        return {"message": "Пароль изменён"}

```

Файл «admin.py»

```

from fastapi import APIRouter, Depends, HTTPException, UploadFile,
File, Query
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.future import select
from sqlalchemy.orm import joinedload
from fastapi.responses import StreamingResponse
from typing import List, Optional
import tempfile, os

from app.core.database import get_db
from app.api.dependencies import get_current_admin
from app.models.user import User
from app.models.student_group import StudentGroup
from app.models.classroom import Classroom
from app.models.discipline import Discipline
from app.models.schedule_item import ScheduleItem
from app.models.teaching_assignment import TeachingAssignment
from app.schemas import admin_schemas
import app.schemas.teacher_schemas as teacher_schemas
from app.services.docx_parser import parse_assignments
from app.services.save_assignments import save_parsed_data
from app.services.schedule_validator import validate_schedule_item
from app.services.excel_export import generate_schedule_excel,
generate_student_schedule_excel
from app.services.schedule_utils import get_teacher_display_name,
enrich_items_with_co_teachers

router = APIRouter(prefix="/api/admin", tags=["admin"])

# ----- Groups -----
@router.get("/groups", response_model=List[admin_schemas.GroupOut])
async def list_groups(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    result = await db.execute(select(StudentGroup))
    return result.scalars().all()

@router.post("/groups", response_model=admin_schemas.GroupOut)
async def create_group(group: admin_schemas.GroupCreate, db:
AsyncSession = Depends(get_db), current_user: User =
Depends(get_current_admin)):
    db_group = StudentGroup(**group.model_dump())
    db.add(db_group)

```

```

        await db.commit()
        await db.refresh(db_group)
        return db_group

@router.put("/groups/{group_id}",
response_model=admin_schemas.GroupOut)
async def update_group(group_id: int, group:
admin_schemas.GroupUpdate, db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    res = await
db.execute(select(StudentGroup).where(StudentGroup.id == group_id))
    db_group = res.scalar_one_or_none()
    if not db_group:
        raise HTTPException(status_code=404, detail="Group not
found")
    for k, v in group.model_dump(exclude_unset=True).items():
        setattr(db_group, k, v)
    await db.commit()
    return db_group

@router.delete("/groups/{group_id}")
async def delete_group(group_id: int, db: AsyncSession =
Depends(get_db), current_user: User = Depends(get_current_admin)):
    res = await
db.execute(select(StudentGroup).where(StudentGroup.id == group_id))
    db_group = res.scalar_one_or_none()
    if not db_group:
        raise HTTPException(status_code=404, detail="Group not
found")
    await db.delete(db_group)
    await db.commit()
    return {"message": "Group deleted"}

# ----- Classrooms -----
@router.get("/classrooms",
response_model=List[admin_schemas.ClassroomOut])
async def list_classrooms(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    result = await db.execute(select(Classroom))
    return result.scalars().all()

@router.post("/classrooms",
response_model=admin_schemas.ClassroomOut)
async def create_classroom(classroom: admin_schemas.ClassroomCreate,
db: AsyncSession = Depends(get_db), current_user: User =
Depends(get_current_admin)):
    db_classroom = Classroom(**classroom.model_dump())
    db.add(db_classroom)
    await db.commit()
    await db.refresh(db_classroom)
    return db_classroom

```

```

@router.put("/classrooms/{classroom_id}",
response_model=admin_schemas.ClassroomOut)
async def update_classroom(classroom_id: int, classroom:
admin_schemas.ClassroomUpdate, db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    res = await db.execute(select(Classroom).where(Classroom.id ==
classroom_id))
    db_classroom = res.scalar_one_or_none()
    if not db_classroom:
        raise HTTPException(status_code=404, detail="Classroom not
found")
    for k, v in classroom.model_dump(exclude_unset=True).items():
        setattr(db_classroom, k, v)
    await db.commit()
    return db_classroom

@router.delete("/classrooms/{classroom_id}")
async def delete_classroom(classroom_id: int, db: AsyncSession =
Depends(get_db), current_user: User = Depends(get_current_admin)):
    res = await db.execute(select(Classroom).where(Classroom.id ==
classroom_id))
    db_classroom = res.scalar_one_or_none()
    if not db_classroom:
        raise HTTPException(status_code=404, detail="Classroom not
found")
    await db.delete(db_classroom)
    await db.commit()
    return {"message": "Classroom deleted"}

# ----- Disciplines -----
@router.get("/disciplines",
response_model=List[admin_schemas.DisciplineOut])
async def list_disciplines(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    result = await db.execute(select(Discipline))
    return result.scalars().all()

@router.post("/disciplines",
response_model=admin_schemas.DisciplineOut)
async def create_discipline(discipline:
admin_schemas.DisciplineCreate, db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    db_discipline = Discipline(**discipline.model_dump())
    db.add(db_discipline)
    await db.commit()
    await db.refresh(db_discipline)
    return db_discipline

@router.put("/disciplines/{discipline_id}",
response_model=admin_schemas.DisciplineOut)
async def update_discipline(discipline_id: int, discipline:
admin_schemas.DisciplineUpdate, db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):

```

```

        res = await db.execute(select(Discipline).where(Discipline.id ==
discipline_id))
        db_discipline = res.scalar_one_or_none()
        if not db_discipline:
            raise HTTPException(status_code=404, detail="Discipline not
found")
        for k, v in discipline.model_dump(exclude_unset=True).items():
            setattr(db_discipline, k, v)
        await db.commit()
        return db_discipline

@router.delete("/disciplines/{discipline_id}")
async def delete_discipline(discipline_id: int, db: AsyncSession =
Depends(get_db), current_user: User = Depends(get_current_admin)):
    res = await db.execute(select(Discipline).where(Discipline.id ==
discipline_id))
    db_discipline = res.scalar_one_or_none()
    if not db_discipline:
        raise HTTPException(status_code=404, detail="Discipline not
found")
    await db.delete(db_discipline)
    await db.commit()
    return {"message": "Discipline deleted"}

# ----- Teachers (Users) -----
@router.get("/teachers",
response_model=List[admin_schemas.TeacherOut])
async def list_teachers(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    result = await db.execute(select(User).where(User.role ==
"teacher"))
    return result.scalars().all()

@router.post("/teachers", response_model=admin_schemas.TeacherOut)
async def create_teacher(teacher: admin_schemas.TeacherCreate, db:
AsyncSession = Depends(get_db), current_user: User =
Depends(get_current_admin)):
    exist = await db.execute(select(User).where(User.username ==
teacher.username))
    if exist.scalar_one_or_none():
        raise HTTPException(status_code=400, detail="Username
already exists")
    from app.core.security import hash_password
    user = User(
        username=teacher.username,
        password_hash=hash_password(teacher.password),
        full_name=teacher.full_name,
        role="teacher",
        academic_degree=teacher.academic_degree,
        department=teacher.department
    )
    db.add(user)
    await db.commit()

```

```

        await db.refresh(user)
        return user

# ----- Upload assignments -----
@router.post("/upload-assignments")
async def upload_assignments(
    file: UploadFile = File(...),
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    if not file.filename.endswith('.docx'):
        raise HTTPException(status_code=400, detail="Только .docx
        файлы поддерживаются")

    with tempfile.NamedTemporaryFile(delete=False, suffix='.docx')
    as tmp:
        content = await file.read()
        tmp.write(content)
        tmp_path = tmp.name

    try:
        data = parse_assignments(tmp_path)
        stats = await save_parsed_data(data, db)
        return {
            "status": "success",
            "stats": stats,
            "summary": {
                "groups_count": len(data['groups']),
                "classrooms_count": len(data['classrooms']),
                "disciplines_count": len(data['disciplines']),
                "teachers_count": len(data['teachers']),
                "assignments_count": len(data['assignments']),
            }
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Ошибка
        парсинга: {str(e)}")
    finally:
        os.unlink(tmp_path)

# ----- Schedule approval -----
@router.get("/pending-schedules")
async def pending_schedules(db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)):
    subq = select(ScheduleItem.created_by).where(ScheduleItem.status
    == "pending").distinct()
    res = await db.execute(subq)
    teacher_ids = res.scalars().all()
    if not teacher_ids:
        return []
    users = await
    db.execute(select(User).where(User.id.in_(teacher_ids)))
    return users.scalars().all()

```

```

@router.post("/approve/{teacher_user_id}")
async def approve_schedule(
    teacher_user_id: int,
    db: AsyncSession = Depends(get_db) ,
    current_user: User = Depends(get_current_admin)
):
    # Находим все pending занятия для этого преподавателя (основные)
    items = await db.execute(
        select(ScheduleItem)
        .join(TeachingAssignment)
        .where(
            TeachingAssignment.user_id == teacher_user_id,
            ScheduleItem.status == "pending"
        )
    )
    items = items.scalars().all()

    # Собираем все связанные записи (мастер + приглашения)
    all_to_approve = set()
    for item in items:
        if item.collaboration_master_id is None:
            # Мастер
            all_to_approve.add(item)
            invites = await db.execute(

select(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
item.id)

            )
            for inv in invites.scalars().all():
                all_to_approve.add(inv)
        else:
            # Приглашение - добавляем его и мастера
            all_to_approve.add(item)
            master = await db.get(ScheduleItem,
item.collaboration_master_id)
            if master:
                all_to_approve.add(master)

    for item in all_to_approve:
        item.status = "approved"

    await db.commit()
    return {"message": f"Утверждено {len(all_to_approve)} занятий
(включая совместные)"}

@router.post("/reject/{teacher_user_id}")
async def reject_schedule(
    teacher_user_id: int,
    reason: str = "",
    propose_edits: bool = False,
    db: AsyncSession = Depends(get_db) ,
    current_user: User = Depends(get_current_admin)

```

```

):
    # Находим все pending занятия для этого преподавателя
    items = await db.execute(
        select(ScheduleItem)
        .join(TeachingAssignment)
        .where(
            TeachingAssignment.user_id == teacher_user_id,
            ScheduleItem.status == "pending"
        )
    )
    items = items.scalars().all()

    all_to_reject = set()
    for item in items:
        if item.collaboration_master_id is None:
            all_to_reject.add(item)
            invites = await db.execute(
                select(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
item.id)
            )
            for inv in invites.scalars().all():
                all_to_reject.add(inv)
        else:
            all_to_reject.add(item)
            master = await db.get(ScheduleItem,
item.collaboration_master_id)
            if master:
                all_to_reject.add(master)

    for item in all_to_reject:
        item.status = "draft"
        item.rejection_reason = reason
        item.proposed_by_admin = "pending" if propose_edits else
None

    await db.commit()
    return {"message": f"Отклонено {len(all_to_reject)} занятий"}

# ----- Просмотр расписания преподавателя -----
@router.get("/schedule/{teacher_user_id}",
response_model=List[teacher_schemas.ScheduleItemOut])
async def get_teacher_schedule(
    teacher_user_id: int,
    week_type: Optional[str] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    teacher = await db.get(User, teacher_user_id)
    if not teacher or teacher.role != "teacher":
        raise HTTPException(status_code=404, detail="Преподаватель
не найден")

```

```

# 1. Занятия, где преподаватель является основным (user_id)
owner_query = (
    select(ScheduleItem)
    .join(TeachingAssignment)
    .where(
        TeachingAssignment.user_id == teacher_user_id,
        ScheduleItem.invited_user_id == None
    )
    .options(
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.discipline),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.group),
        joinedload(ScheduleItem.classroom),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.teacher)
    )
)

# 2. Занятия, где преподаватель приглашён (invited_user_id)
invited_query = (
    select(ScheduleItem)
    .join(TeachingAssignment,
ScheduleItem.teaching_assignment_id == TeachingAssignment.id)
    .where(ScheduleItem.invited_user_id == teacher_user_id)
    .options(
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.discipline),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.group),
        joinedload(ScheduleItem.classroom),
        joinedload(ScheduleItem.teaching_assignment)
        .joinedload(TeachingAssignment.teacher)
    )
)

if week_type in ("odd", "even"):
    owner_query =
owner_query.where(ScheduleItem.week_type.in_([week_type, "all"]))
    invited_query =
invited_query.where(ScheduleItem.week_type.in_([week_type, "all"]))

    owner_items = (await
db.execute(owner_query)).unique().scalars().all()
    invited_items = (await
db.execute(invited_query)).unique().scalars().all()
    all_items = list(owner_items) + list(invited_items)

# Получаем имена преподавателей (с соавторами) для каждого
элемента
    teacher_name_map = await enrich_items_with_co_teachers(db,
all_items)

```

```

result = []
for item in all_items:
    ta = item.teaching_assignment
    result.append({
        "id": item.id,
        "teaching_assignment_id": item.teaching_assignment_id,
        "classroom_id": item.classroom_id,
        "day_of_week": item.day_of_week,
        "pair_number": item.pair_number,
        "week_type": item.week_type,
        "status": item.status,
        "created_at": item.created_at,
        "discipline_name": ta.discipline.name if ta.discipline
    else "",
        "group_name": ta.group.name if ta.group else "",
        "lesson_type": ta.lesson_type,
        "teacher_name": teacher_name_map.get(item.id, ""),
        "classroom_number": item.classroom.number if
item.classroom else "",
        "submit_note": item.submit_note,
        "rejection_reason": item.rejection_reason,
        "proposed_by_admin": item.proposed_by_admin,
        # для совместимости со схемой добавляем поля, которые
могут быть нужны фронту
        "invited_user_id": item.invited_user_id,
        "collaboration_master_id": item.collaboration_master_id,
        "collaboration_history": item.collaboration_history or
    [],
    })
return result

# ----- Редактирование занятия администратором -----
@router.put("/schedule-items/{item_id}",
response_model=teacher_schemas.ScheduleItemOut)
async def admin_update_schedule_item(
    item_id: int,
    item_update: teacher_schemas.ScheduleItemCreate,
    propose: bool = False,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    db_item = await db.get(ScheduleItem, item_id)
    if not db_item:
        raise HTTPException(status_code=404, detail="Занятие не
найдено")

    if db_item.status not in ("draft", "pending"):
        raise HTTPException(status_code=400, detail="Разрешено
редактировать только черновики или ожидающие утверждения")

    errors = await validate_schedule_item(
        db, item_update.teaching_assignment_id,
        item_update.classroom_id,

```

```

        item_update.day_of_week, item_update.pair_number,
        exclude_item_id=item_id
    )
    if errors:
        raise HTTPException(status_code=400, detail=";
".join(errors))

    db_item.teaching_assignment_id =
item_update.teaching_assignment_id
    db_item.classroom_id = item_update.classroom_id
    db_item.day_of_week = item_update.day_of_week
    db_item.pair_number = item_update.pair_number
    db_item.week_type = item_update.week_type or "all"

    if propose:
        db_item.proposed_by_admin = 'pending'
        # статус сохраняется, чтобы преподаватель увидел изменение

    await db.commit()
    return db_item

# ----- Удаление занятия администратором -----
@router.delete("/schedule-items/{item_id}")
async def admin_delete_schedule_item(
    item_id: int,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    db_item = await db.get(ScheduleItem, item_id)
    if not db_item:
        raise HTTPException(status_code=404, detail="Занятие не
найдено")

    # Если это мастер-запись (collaboration_master_id is None),
удаляем все приглашения
    if db_item.collaboration_master_id is None:
        from sqlalchemy import delete
        await db.execute(

delete(ScheduleItem).where(ScheduleItem.collaboration_master_id ==
db_item.id)
        )

    await db.delete(db_item)
    await db.commit()
    return {"message": "Занятие удалено"}

# ----- Создание занятия администратором -----
@router.post("/schedule-items",
response_model=teacher_schemas.ScheduleItemOut)
async def admin_create_schedule_item(
    item: teacher_schemas.ScheduleItemCreate,
    teacher_user_id: int = Query(...),

```

```

        propose: bool = False,
        db: AsyncSession = Depends(get_db),
        current_user: User = Depends(get_current_admin)
    ):
        teacher = await db.get(User, teacher_user_id)
        if not teacher or teacher.role != "teacher":
            raise HTTPException(status_code=404, detail="Преподаватель
не найден")

        assignment = await db.get(TeachingAssignment,
item.teaching_assignment_id)
        if not assignment:
            raise HTTPException(status_code=404, detail="Назначение не
найдено")

        if assignment.user_id != teacher_user_id:
            raise HTTPException(status_code=403, detail="Назначение не
принадлежит указанному преподавателю")

        errors = await validate_schedule_item(
            db, item.teaching_assignment_id, item.classroom_id,
            item.day_of_week, item.pair_number
        )
        if errors:
            raise HTTPException(status_code=400, detail=";
".join(errors))

        schedule_item = ScheduleItem(
            teaching_assignment_id=item.teaching_assignment_id,
            classroom_id=item.classroom_id,
            day_of_week=item.day_of_week,
            pair_number=item.pair_number,
            week_type=item.week_type or "all",
            status="draft",
            created_by=teacher_user_id
        )
        if propose:
            schedule_item.proposed_by_admin = 'pending'

        db.add(schedule_item)
        await db.commit()
        await db.refresh(schedule_item)
        return schedule_item

# ----- Назначения преподавателя -----
@router.get("/teacher/{teacher_user_id}/assignments",
response_model=List[teacher_schemas.TeacherAssignmentOut])
async def get_teacher_assignments(
    teacher_user_id: int,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    teacher = await db.get(User, teacher_user_id)

```

```

        if not teacher or teacher.role != "teacher":
            raise HTTPException(status_code=404, detail="Преподаватель
не найден")

        assignments = await db.execute(
            select(TeachingAssignment)
                .options(joinedload(TeachingAssignment.discipline),
joinedload(TeachingAssignment.group))
                .where(TeachingAssignment.user_id == teacher_user_id)
        )
        assignments = assignments.unique().scalars().all()
        return [teacher_schemas.TeacherAssignmentOut.from_orm(a) for a
in assignments]

# ----- Доступные аудитории (админ) -----
@router.get("/available-slots",
response_model=teacher_schemas.AvailableSlotsResponse)
async def admin_available_slots(
    teaching_assignment_id: int = Query(...),
    day_of_week: int = Query(...),
    pair_number: int = Query(...),
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    assignment = await db.get(TeachingAssignment,
teaching_assignment_id)
    if not assignment:
        raise HTTPException(status_code=404, detail="Назначение не
найдено")

    errors = await validate_schedule_item(
        db, teaching_assignment_id, classroom_id=0,
        day_of_week=day_of_week, pair_number=pair_number
    )
    slot_free = not any(e for e in errors if "аудитория" not in
e.lower())

    lesson_type = assignment.lesson_type
    all_rooms_q = select(Classroom)
    all_rooms = (await db.execute(all_rooms_q)).scalars().all()

    free_classrooms = []
    for room in all_rooms:
        # Проверяем, есть ли хотя бы одно занятие, занимающее эту
аудиторию в данный слот
        conflict_query = select(ScheduleItem).where(
            ScheduleItem.classroom_id == room.id,
            ScheduleItem.day_of_week == day_of_week,
            ScheduleItem.pair_number == pair_number,
            ScheduleItem.status.in_(["pending", "approved"])
        ).limit(1) # нам достаточно узнать, есть ли хоть одно
        conflict = await db.execute(conflict_query)
        if conflict.scalar_one_or_none() is None: # нет конфликта

```

```

        suitable = True
        if lesson_type == "lecture" and not room.has_projector:
            suitable = False
        elif lesson_type in ("lab", "practice") and not
room.has_computers:
            suitable = False
        free_classrooms.append({
            "id": room.id,
            "number": room.number,
            "has_projector": room.has_projector,
            "has_computers": room.has_computers,
            "suitable": suitable,
        })

    return teacher_schemas.AvailableSlotsResponse(
        slot_free=slot_free,
        free_classrooms=free_classrooms
    )

# ----- JSON-данные для общей ведомости (админ) -----
@router.get("/schedule-data")
async def get_schedule_data(
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    base_query = (
        select(ScheduleItem)
        .options(
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.discipline),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.group),
            joinedload(ScheduleItem.teaching_assignment)
            .joinedload(TeachingAssignment.teacher),
            joinedload(ScheduleItem.classroom),
        )
        .where(ScheduleItem.status == 'approved')
    )
    result = await db.execute(base_query)
    items = result.unique().scalars().all()

    # Получаем имена преподавателей для каждого элемента
    teacher_name_map = await enrich_items_with_co_teachers(db,
items)

    teacher_schedule = {}
    for item in items:
        teacher = item.teaching_assignment.teacher
        if not teacher:
            continue
        t_id = teacher.id
        if t_id not in teacher_schedule:

```

```

        teacher_schedule[t_id] = {'teacher': teacher, 'items':
[]}
        teacher_schedule[t_id]['items'].append(item)

        days = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница',
'суббота']
        time_slots = ['8.00-9.35', '9.50-11.25', '11.40-13.15',
'13.45-15.20', '15.35-17.10', '17.25-19.00',
'19.15-20.50']

        rows = []
        for idx, (t_id, t_data) in enumerate(teacher_schedule.items(),
start=1):
            teacher = t_data['teacher']
            row_data = {'num': idx, 'teacher_name': teacher.full_name,
'cells': {}}
            for item in t_data['items']:
                day_idx = item.day_of_week - 1
                if 0 <= day_idx < len(days):
                    pair_idx = item.pair_number - 1
                    if 0 <= pair_idx < 7:
                        key = f"{day_idx}-{pair_idx}"
                        week_note = ''
                        if item.week_type == 'odd':
                            week_note = ' (неч.)'
                        elif item.week_type == 'even':
                            week_note = ' (чет.)'
                        teacher_names = teacher_name_map.get(item.id,
teacher.full_name)
                        info = (f"{item.teaching_assignment.group.name
if item.teaching_assignment.group else ''} "
f"{item.teaching_assignment.discipline.name if
item.teaching_assignment.discipline else ''} "
f"{'лк.' if
item.teaching_assignment.lesson_type == 'lecture' else 'лб.'} "
f"{teacher_names} "
f"ауд.{item.classroom.number if
item.classroom else ''}"
f"{week_note}")
                        row_data['cells'][key] = info
            rows.append(row_data)

        return {'days': days, 'time_slots': time_slots, 'rows': rows}

# ----- JSON-данные для студенческой ведомости -----
@router.get("/student-schedule-data")
async def get_student_schedule_data(
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    base_query = (
        select(ScheduleItem)

```



```

        item_all = next((it for it in g['items']
                        if it.day_of_week == day_num
                        and it.pair_number == pair_num
                        and it.week_type == 'all'), None)

        cell = {'odd_text': None, 'even_text': None,
'all_text': None}
        if item_all:
            disc =
f"{item_all.teaching_assignment.discipline.name if
item_all.teaching_assignment.discipline else ''} ({'лек.' if
item_all.teaching_assignment.lesson_type == 'lecture' else 'лаб.'})"
            teacher = f"{teacher_name_map.get(item_all.id,
'')} ауд.{item_all.classroom.number if item_all.classroom else ''}"
            cell['all_text'] = f"{disc}\n{teacher}"
        else:
            if item_odd:
                disc_odd =
f"{item_odd.teaching_assignment.discipline.name if
item_odd.teaching_assignment.discipline else ''} ({'лек.' if
item_odd.teaching_assignment.lesson_type == 'lecture' else 'лаб.'})"
                t_odd = f"{teacher_name_map.get(item_odd.id,
'')} ауд.{item_odd.classroom.number if item_odd.classroom else ''}"
                cell['odd_text'] = f"{disc_odd}\n{t_odd}"
            if item_even:
                disc_even =
f"{item_even.teaching_assignment.discipline.name if
item_even.teaching_assignment.discipline else ''} ({'лек.' if
item_even.teaching_assignment.lesson_type == 'lecture' else
'лаб.'})"
                t_even =
f"{teacher_name_map.get(item_even.id, '')} ауд.
{item_even.classroom.number if item_even.classroom else ''}"
                cell['even_text'] = f"{disc_even}\n{t_even}"
            cells.append(cell)
            slots.append({'time': slot_time, 'cells': cells})
            days_data.append({'day': day, 'slots': slots})

    return {'days': days_short, 'time_slots': time_slots, 'groups':
groups_info, 'days_data': days_data}

# ----- Экспорт в Excel -----
@router.get("/export-schedule")
async def export_schedule(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    excel_file = await generate_schedule_excel(db)
    return StreamingResponse(
        excel_file,
        media_type="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
        headers={"Content-Disposition": "attachment;
filename=schedule.xlsx"}
    )

```

```

@router.get("/export-student-schedule")
async def export_student_schedule(
    group_id: Optional[int] = None,
    db: AsyncSession = Depends(get_db),
    current_user: User = Depends(get_current_admin)
):
    excel_file = await generate_student_schedule_excel(db, group_id)
    filename = f"student_schedule_group_{group_id}.xlsx" if group_id
    else "student_schedule_all.xlsx"
    return StreamingResponse(
        excel_file,
        media_type="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet",
        headers={"Content-Disposition": f"attachment;
filename={filename}"})
    )

# ----- Управление пользователями -----
@router.get("/users", response_model=List[admin_schemas.UserOut])
async def list_users(db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):
    result = await db.execute(select(User))
    return result.scalars().all()

@router.post("/users", response_model=admin_schemas.UserOut)
async def create_user(user_data: admin_schemas.UserCreate, db:
AsyncSession = Depends(get_db), current_user: User =
Depends(get_current_admin)):
    exist = await db.execute(select(User).where(User.username ==
user_data.username))
    if exist.scalar_one_or_none():
        raise HTTPException(status_code=400, detail="Username
already exists")
    from app.core.security import hash_password
    user = User(
        username=user_data.username,
        password_hash=hash_password(user_data.password),
        full_name=user_data.full_name,
        role=user_data.role,
        academic_degree=user_data.academic_degree,
        department=user_data.department
    )
    db.add(user)
    await db.commit()
    await db.refresh(user)
    return user

@router.put("/users/{user_id}",
response_model=admin_schemas.UserOut)
async def update_user(user_id: int, user_data:
admin_schemas.UserUpdate, db: AsyncSession = Depends(get_db),
current_user: User = Depends(get_current_admin)):

```

```

        user = await db.get(User, user_id)
        if not user:
            raise HTTPException(status_code=404, detail="User not found")
        if user_data.username is not None:
            exist = await db.execute(select(User).where(User.username == user_data.username, User.id != user_id))
            if exist.scalar_one_or_none():
                raise HTTPException(status_code=400, detail="Username already taken")
            user.username = user_data.username
        if user_data.full_name is not None:
            user.full_name = user_data.full_name
        if user_data.role is not None:
            user.role = user_data.role
        if user_data.academic_degree is not None:
            user.academic_degree = user_data.academic_degree
        if user_data.department is not None:
            user.department = user_data.department
        if user_data.password is not None:
            from app.core.security import hash_password
            user.password_hash = hash_password(user_data.password)
        await db.commit()
        await db.refresh(user)
        return user

@router.delete("/users/{user_id}")
async def delete_user(user_id: int, db: AsyncSession = Depends(get_db), current_user: User = Depends(get_current_admin)):
    user = await db.get(User, user_id)
    if not user:
        raise HTTPException(status_code=404, detail="User not found")
    if user.role == 'admin':
        from sqlalchemy.sql import func
        admin_count = await db.scalar(select(func.count()).select_from(User).where(User.role == 'admin'))
        if admin_count <= 1 and user.id == current_user.id:
            raise HTTPException(status_code=400, detail="Нельзя удалить последнего администратора")
        await db.delete(user)
        await db.commit()
        return {"message": "Пользователь удалён"}

```

Файл «index.html»

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.svg" />

```

```

    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>university-schedule-frontend</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>

```

Файл «package.json»

```

{
  "name": "university-schedule-frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.14.0",
    "@emotion/styled": "^11.14.1",
    "@fontsource/roboto": "^5.2.10",
    "@mui/material": "^9.0.0",
    "axios": "^1.16.0",
    "react": "^19.2.5",
    "react-dom": "^19.2.5",
    "react-router-dom": "^7.14.2"
  },
  "devDependencies": {
    "@eslint/js": "^9.39.4",
    "@types/react": "^19.2.14",
    "@types/react-dom": "^19.2.3",
    "@vitejs/plugin-react": "^6.0.1",
    "eslint": "^9.39.4",
    "eslint-plugin-react-hooks": "^7.1.1",
    "eslint-plugin-react-refresh": "^0.5.2",
    "globals": "^17.5.0",
    "typescript": "^6.0.3",
    "vite": "^8.0.9"
  }
}

```

Файл «main.jsx»

```

import React from 'react';
import ReactDOM from 'react-dom/client';

```

```
import App from './App';
import '@fontsource/roboto/300.css';
import '@fontsource/roboto/400.css';
import '@fontsource/roboto/500.css';
import '@fontsource/roboto/700.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Файл «index.css»

```
:root {
  --text: #6b6375;
  --text-h: #08060d;
  --bg: #fff;
  --border: #e5e4e7;
  --code-bg: #f4f3ec;
  --accent: #aa3bff;
  --accent-bg: rgba(170, 59, 255, 0.1);
  --accent-border: rgba(170, 59, 255, 0.5);
  --social-bg: rgba(244, 243, 236, 0.5);
  --shadow:
    rgba(0, 0, 0, 0.1) 0 10px 15px -3px, rgba(0, 0, 0, 0.05) 0 4px
    6px -2px;

  --sans: system-ui, 'Segoe UI', Roboto, sans-serif;
  --heading: system-ui, 'Segoe UI', Roboto, sans-serif;
  --mono: ui-monospace, Consolas, monospace;

  font: 18px/145% var(--sans);
  letter-spacing: 0.18px;
  color-scheme: light dark;
  color: var(--text);
  background: var(--bg);
  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;

  @media (max-width: 1024px) {
    font-size: 16px;
  }
}

@media (prefers-color-scheme: dark) {
  :root {
    --text: #9ca3af;
    --text-h: #f3f4f6;
    --bg: #16171d;
    --border: #2e303a;
    --code-bg: #1f2028;
```

```

    --accent: #c084fc;
    --accent-bg: rgba(192, 132, 252, 0.15);
    --accent-border: rgba(192, 132, 252, 0.5);
    --social-bg: rgba(47, 48, 58, 0.5);
    --shadow:
        rgba(0, 0, 0, 0.4) 0 10px 15px -3px, rgba(0, 0, 0, 0.25) 0 4px
6px -2px;
}

#social .button-icon {
    filter: invert(1) brightness(2);
}
}

body {
    margin: 0;
}

#root {
    width: 1126px;
    max-width: 100%;
    margin: 0 auto;
    text-align: center;
    border-inline: 1px solid var(--border);
    min-height: 100svh;
    display: flex;
    flex-direction: column;
    box-sizing: border-box;
}

h1,
h2 {
    font-family: var(--heading);
    font-weight: 500;
    color: var(--text-h);
}

h1 {
    font-size: 56px;
    letter-spacing: -1.68px;
    margin: 32px 0;
    @media (max-width: 1024px) {
        font-size: 36px;
        margin: 20px 0;
    }
}

h2 {
    font-size: 24px;
    line-height: 118%;
    letter-spacing: -0.24px;
    margin: 0 0 8px;
    @media (max-width: 1024px) {
        font-size: 20px;
    }
}

```

```

    }
  }
  p {
    margin: 0;
  }

  code,
  .counter {
    font-family: var(--mono);
    display: inline-flex;
    border-radius: 4px;
    color: var(--text-h);
  }

  code {
    font-size: 15px;
    line-height: 135%;
    padding: 4px 8px;
    background: var(--code-bg);
  }

```

Файл «App.jsx»

```

import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import { AuthProvider, useAuth } from '../context/AuthContext';
import LoginPage from '../pages/LoginPage';
import TeacherPage from '../pages/TeacherPage';
import AdminPage from '../pages/AdminPage';
import { CircularProgress, Box } from '@mui/material';

// Компонент для защищённых маршрутов
const ProtectedRoute = ({ children, role }) => {
  const { user, loading } = useAuth();

  if (loading) {
    return (
      <Box sx={{ display: 'flex', justifyContent: 'center', mt:
10 }}>
        <CircularProgress />
      </Box>
    );
  }

  if (!user) {
    return <Navigate to="/login" replace />;
  }

  if (role && user.role !== role) {
    return <Navigate to="/login" replace />;
  }

  return children;
};

```

```

function AppRoutes() {
  const { user, loading } = useAuth();

  if (loading) {
    return (
      <Box sx={{ display: 'flex', justifyContent: 'center', mt:
10 }}>
        <CircularProgress />
      </Box>
    );
  }

  return (
    <Routes>
      <Route path="/login" element={<LoginPage />} />
      <Route
        path="/teacher/*"
        element={
          <ProtectedRoute role="teacher">
            <TeacherPage />
          </ProtectedRoute>
        }
      />
      <Route
        path="/admin/*"
        element={
          <ProtectedRoute role="admin">
            <AdminPage />
          </ProtectedRoute>
        }
      />
      { /* Редирект с корня на страницу входа или на личный кабинет
*/ }
      <Route
        path="/"
        element={
          user ? (
            <Navigate to={user.role === 'admin' ? '/admin' :
'/' + user.role} replace />
          ) : (
            <Navigate to="/login" replace />
          )
        }
      />
    </Routes>
  );
}

export default function App() {
  return (
    <BrowserRouter>
      <AuthProvider>

```

```

        <AppRoutes />
      </AuthProvider>
    </BrowserRouter>
  );
}

```

Файл «App.css»

```

.counter {
  font-size: 16px;
  padding: 5px 10px;
  border-radius: 5px;
  color: var(--accent);
  background: var(--accent-bg);
  border: 2px solid transparent;
  transition: border-color 0.3s;
  margin-bottom: 24px;

  &:hover {
    border-color: var(--accent-border);
  }
  &:focus-visible {
    outline: 2px solid var(--accent);
    outline-offset: 2px;
  }
}

.hero {
  position: relative;

  .base,
  .framework,
  .vite {
    inset-inline: 0;
    margin: 0 auto;
  }

  .base {
    width: 170px;
    position: relative;
    z-index: 0;
  }

  .framework,
  .vite {
    position: absolute;
  }

  .framework {
    z-index: 1;
    top: 34px;
    height: 28px;
    transform: perspective(2000px) rotateZ(300deg) rotateX(44deg)
    rotateY(39deg)

```

```

        scale(1.4);
    }

    .vite {
        z-index: 0;
        top: 107px;
        height: 26px;
        width: auto;
        transform: perspective(2000px) rotateZ(300deg) rotateX(40deg)
rotateY(39deg)
        scale(0.8);
    }
}

#center {
    display: flex;
    flex-direction: column;
    gap: 25px;
    place-content: center;
    place-items: center;
    flex-grow: 1;

    @media (max-width: 1024px) {
        padding: 32px 20px 24px;
        gap: 18px;
    }
}

#next-steps {
    display: flex;
    border-top: 1px solid var(--border);
    text-align: left;

    & > div {
        flex: 1 1 0;
        padding: 32px;
        @media (max-width: 1024px) {
            padding: 24px 20px;
        }
    }

    .icon {
        margin-bottom: 16px;
        width: 22px;
        height: 22px;
    }

    @media (max-width: 1024px) {
        flex-direction: column;
        text-align: center;
    }
}

```

```

#docs {
  border-right: 1px solid var(--border);

  @media (max-width: 1024px) {
    border-right: none;
    border-bottom: 1px solid var(--border);
  }
}

#next-steps ul {
  list-style: none;
  padding: 0;
  display: flex;
  gap: 8px;
  margin: 32px 0 0;

  .logo {
    height: 18px;
  }

  a {
    color: var(--text-h);
    font-size: 16px;
    border-radius: 6px;
    background: var(--social-bg);
    display: flex;
    padding: 6px 12px;
    align-items: center;
    gap: 8px;
    text-decoration: none;
    transition: box-shadow 0.3s;

    &:hover {
      box-shadow: var(--shadow);
    }

    .button-icon {
      height: 18px;
      width: 18px;
    }
  }
}

@media (max-width: 1024px) {
  margin-top: 20px;
  flex-wrap: wrap;
  justify-content: center;

  li {
    flex: 1 1 calc(50% - 8px);
  }

  a {
    width: 100%;
    justify-content: center;
  }
}

```

```

        box-sizing: border-box;
    }
}

#spacer {
    height: 88px;
    border-top: 1px solid var(--border);
    @media (max-width: 1024px) {
        height: 48px;
    }
}

.ticks {
    position: relative;
    width: 100%;

    &::before,
    &::after {
        content: '';
        position: absolute;
        top: -4.5px;
        border: 5px solid transparent;
    }

    &::before {
        left: 0;
        border-left-color: var(--border);
    }
    &::after {
        right: 0;
        border-right-color: var(--border);
    }
}

```

Файл «TeacherPage.jsx»

```

import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import {
    Box, Typography, Button, FormControl, InputLabel, Select,
    MenuItem, TextField, Alert, Tabs, Tab
} from '@mui/material';
import { useAuth } from '../context/AuthContext';
import ScheduleGrid from '../components/ScheduleGrid';
import SlotModal from '../components/SlotModal';
import { submitSchedule, getMySchedule, clearProposed } from
    '../api/teacherApi';
import TeacherScheduleView from '../components/TeacherScheduleView';
import axiosInstance from '../api/axiosInstance';

export default function TeacherPage() {
    const { user, logout } = useAuth();
    const navigate = useNavigate();

```

```

    const [tab, setTab] = useState(0); // 0-редактирование, 1-моя
ведомость
    const [modalOpen, setModalOpen] = useState(false);
    const [selectedDay, setSelectedDay] = useState(null);
    const [selectedPair, setSelectedPair] = useState(null);
    const [editingItem, setEditingItem] = useState(null);
    const [editModalOpen, setEditModalOpen] = useState(false);
    const [refreshKey, setRefreshKey] = useState(0);
    const [weekOffset, setWeekOffset] = useState(0);
    const semesterStart = new Date(2026, 1, 9);
    const weekType = weekOffset % 2 === 0 ? 'even' : 'odd';
    const [submitNote, setSubmitNote] = useState('');
    const [proposedExists, setProposedExists] = useState(false);

    const getDatesForWeek = (offset) => {
        const start = new Date(semesterStart);
        start.setDate(start.getDate() + offset * 7);
        const dates = [];
        for (let i = 0; i < 6; i++) {
            const day = new Date(start);
            day.setDate(day.getDate() + i);
            dates.push(day.toLocaleDateString('ru-RU', { day: 'numeric',
month: 'short' }));
        }
        return dates;
    };

    const weekDates = getDatesForWeek(weekOffset);

    useEffect(() => {
        getMySchedule(weekType).then(data => {
            setProposedExists(data.some(item => item.proposed_by_admin));
        });
    }, [refreshKey, weekType]);

    const handleCellClick = (day, pair) => {
        setSelectedDay(day);
        setSelectedPair(pair);
        setModalOpen(true);
    };

    const handleDraftClick = (item) => {
        setEditingItem(item);
        setEditModalOpen(true);
    };

    const handleSaved = () => {
        setRefreshKey(prev => prev + 1);
        setEditingItem(null);
    };

    const handleLogout = () => {
        logout();
        navigate('/login');
    };

```

```

};

const handleSubmit = async () => {
  await submitSchedule(submitNote);
  setRefreshKey(prev => prev + 1);
  setSubmitNote('');
};

const handleAcceptProposed = async () => {
  await submitSchedule('Предложенные правки приняты');
  setRefreshKey(prev => prev + 1);
};

const handleRejectProposed = async () => {
  await clearProposed();
  setRefreshKey(prev => prev + 1);
};

const handleWeekChange = (e) => {
  setWeekOffset(Number(e.target.value));
  setRefreshKey(prev => prev + 1);
};

// Экспорт личной ведомости преподавателя
const handleExportMySchedule = async () => {
  try {
    const response = await axiosInstance.get('/api/teacher/export-my-schedule', { responseType: 'blob' });
    const url = window.URL.createObjectURL(new Blob([response.data]));
    const link = document.createElement('a');
    link.href = url;
    link.setAttribute('download', 'my_schedule.xlsx');
    document.body.appendChild(link);
    link.click();
    link.remove();
  } catch (error) {
    console.error('Ошибка экспорта', error);
  }
};

return (
  <Box sx={{ width: '100%', px: 4, mt: 4 }}>
    <Typography variant="h4" gutterBottom>
      Добро пожаловать, {user?.full_name}!
    </Typography>

    <Tabs value={tab} onChange={(e, newVal) => setTab(newVal)}
    sx={{ mb: 2 }}>
      <Tab label="Редактирование" />
      <Tab label="Общая ведомость" />
    </Tabs>
  </Box>
);

```

```

{tab === 0 && (
  <>
    {proposedExists && (
      <Alert severity="info" sx={{ mb: 2 }}>
        Администратор предложил исправленное расписание.
      </Alert>
    )}

    <Box sx={{ mb: 2, display: 'flex', alignItems: 'center',
flexWrap: 'wrap', gap: 2 }}>
      <Button variant="contained"
onClick={handleSubmit}>Отправить на утверждение</Button>
      {proposedExists && (
        <>
          <Button variant="outlined" color="success"
onClick={handleAcceptProposed}>Принять предложенные правки</Button>
          <Button variant="outlined" color="warning"
onClick={handleRejectProposed}>Отклонить предложение</Button>
        </>
      )}
      <Button variant="outlined" color="error"
onClick={handleLogout}>Выйти</Button>

      <TextField
        label="Примечание к отправке"
        value={submitNote}
        onChange={(e) => setSubmitNote(e.target.value)}
        size="small"
        sx={{ minWidth: 250 }}
      />

      <FormControl size="small" sx={{ minWidth: 200 }}>
        <InputLabel>Неделя</InputLabel>
        <Select value={weekOffset} label="Неделя"
onChange={handleWeekChange}>
          {Array.from({ length: 18 }, (_, i) => (
            <MenuItem key={i} value={i}>
              {i + 1}-я неделя ({getDatesForWeek(i)[0]} -
{getDatesForWeek(i)[5]})
            </MenuItem>
          )}
        </Select>
      </FormControl>
    </Box>

    <ScheduleGrid
      onClick={handleCellClick}
      onDraftClick={handleDraftClick}
      refreshKey={refreshKey}
      dates={weekDates}
      weekType={weekType}
    />

```

```

        {selectedDay && selectedPair && (
          <SlotModal
            open={modalOpen}
            onClose={() => setModalOpen(false)}
            dayOfWeek={selectedDay}
            pairNumber={selectedPair}
            onSave={handleSaved}
          />
        )}

        {editingItem && (
          <SlotModal
            open={editModalOpen}
            onClose={() => { setEditModalOpen(false);
setEditingItem(null); }}
            editItem={editingItem}
            onSave={handleSaved}
          />
        )}
      </>
    )}

    {tab === 1 && (
      <Box>
        <Box sx={{ mb: 2 }}>
          <Button variant="contained"
onClick={handleExportMySchedule}>
            Скачать ведомость (Excel)
          </Button>
        </Box>
        <TeacherScheduleView />
      </Box>
    )}
  </Box>
);
}

```

Файл «LoginPage.jsx»

```

import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import {
  Container,
  Box,
  TextField,
  Button,
  Typography,
  Alert,
} from '@mui/material';
import { useAuth } from '../context/AuthContext';

export default function LoginPage() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');

```

```

const [error, setError] = useState('');
const { login, user } = useAuth();
const navigate = useNavigate();

// Если пользователь уже авторизован, перенаправляем
if (user) {
  navigate(user.role === 'admin' ? '/admin' : '/teacher',
{ replace: true });
}

const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');
  try {
    await login(username, password);
    // После успешного входа перенаправление произойдёт в
useEffect
  } catch (err) {
    setError(err.response?.data?.detail || 'Ошибка входа');
  }
};

return (
  <Container maxWidth="xs">
    <Box sx={{ mt: 8, display: 'flex', flexDirection: 'column',
alignItems: 'center' }}>
      <Typography component="h1" variant="h5">
        Вход в систему составления расписания
      </Typography>
      {error && <Alert severity="error" sx={{ mt: 2, width: '100%'
}}>{error}</Alert>}
      <Box component="form" onSubmit={handleSubmit} sx={{ mt: 2,
width: '100%' }}>
        <TextField
          margin="normal"
          required
          fullWidth
          label="Имя пользователя"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
        <TextField
          margin="normal"
          required
          fullWidth
          label="Пароль"
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
        <Button type="submit" fullWidth variant="contained"
sx={{ mt: 3, mb: 2 }}>
          Войти

```

```

        </Button>
      </Box>
    </Box>
  </Container>
);
}

```

Файл «AdminPage.jsx»

```

import {
  Container, Typography, Button, Box, Paper, List, ListItem,
  ListItemText,
  Dialog, DialogTitle, DialogContent, DialogActions, TextField,
  FormControl, InputLabel, Select, MenuItem, Tabs, Tab
} from '@mui/material';
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';
import axiosInstance from '../api/axiosInstance';
import { getTeacherSchedule, getTeacherAssignments } from
  '../api/adminApi';
import ScheduleGrid from '../components/ScheduleGrid';
import SlotModal from '../components/SlotModal';
import TeacherScheduleView from '../components/TeacherScheduleView';
import StudentScheduleView from '../components/StudentScheduleView';
import UserManagement from '../components/UserManagement';

export default function AdminPage() {
  const { user, logout } = useAuth();
  const navigate = useNavigate();
  const [tab, setTab] = useState(0);
  const [pendingTeachers, setPendingTeachers] = useState([]);
  const [selectedTeacher, setSelectedTeacher] = useState(null);
  const [scheduleOpen, setScheduleOpen] = useState(false);
  const [teacherSchedule, setTeacherSchedule] = useState([]);
  const [teacherAssignments, setTeacherAssignments] = useState([]);
  const [editMode, setEditMode] = useState(true); // всегда режим
редактирования
  const [weekOffset, setWeekOffset] = useState(0);
  const semesterStart = new Date(2026, 1, 9);
  const weekType = weekOffset % 2 === 0 ? 'even' : 'odd';

  const [editingItem, setEditingItem] = useState(null);
  const [newSlot, setNewSlot] = useState(null);
  const [editModalOpen, setEditModalOpen] = useState(false);

  const [rejectOpen, setRejectOpen] = useState(false);
  const [currentRejectTeacher, setCurrentRejectTeacher] =
useState(null);
  const [rejectReason, setRejectReason] = useState('');
  const [proposeEdits, setProposeEdits] = useState(false);

  const [refreshScheduleKey, setRefreshScheduleKey] = useState(0);

```

```

const handleLogout = () => { logout(); navigate('/login'); };

const fetchPending = async () => {
  try {
    const res = await axiosInstance.get('/api/admin/pending-schedules');
    setPendingTeachers(res.data);
  } catch (err) { console.error(err); }
};

useEffect(() => { fetchPending(); }, []);

const handleApprove = async (teacherId) => {
  await axiosInstance.post(`/api/admin/approve/${teacherId}`);
  fetchPending();
};

const handleRejectClick = (teacher) => {
  setCurrentRejectTeacher(teacher);
  setRejectReason('');
  setProposeEdits(false);
  setRejectOpen(true);
};

const confirmReject = async () => {
  if (currentRejectTeacher) {
    await axiosInstance.post(`/api/admin/reject/${currentRejectTeacher.id}`, null, {
      params: { reason: rejectReason, propose_edits: proposeEdits }
    });
    fetchPending();
  }
  setRejectOpen(false);
};

const openScheduleDialog = async (teacher, editing = true) => {
  setSelectedTeacher(teacher);
  setEditMode(true); // всегда режим редактирования
  try {
    const sched = await getTeacherSchedule(teacher.id, weekType);
    const assigns = await getTeacherAssignments(teacher.id);
    setTeacherSchedule(sched);
    setTeacherAssignments(assigns);
  } catch (err) {
    console.error(err);
    setTeacherSchedule([]);
    setTeacherAssignments([]);
  } finally {
    setScheduleOpen(true);
    setRefreshScheduleKey(prev => prev + 1);
  }
};

```

```

const [proposeToTeacher, setProposeToTeacher] = useState(false);

const handleEditUserSchedule = (teacher) => {
  setProposeToTeacher(true); // все правки будут предложены
  преподавателю
  openScheduleDialog(teacher, true);
};

const handleViewSchedule = (teacher) =>
openScheduleDialog(teacher, false);
// handleEditSchedule теперь всего лишь прокси для
openScheduleDialog (режим редактирования по умолчанию)
const handleEditSchedule = (teacher) =>
openScheduleDialog(teacher, true);

const handleWeekChange = async (e) => {
  const offset = Number(e.target.value);
  setWeekOffset(offset);
  if (selectedTeacher) {
    const newWeekType = offset % 2 === 0 ? 'even' : 'odd';
    const data = await getTeacherSchedule(selectedTeacher.id,
newWeekType);
    setTeacherSchedule(data);
  }
};

const getDatesForWeek = (offset) => {
  const start = new Date(semesterStart);
  start.setDate(start.getDate() + offset * 7);
  const dates = [];
  for (let i = 0; i < 6; i++) {
    const day = new Date(start);
    day.setDate(day.getDate() + i);
    dates.push(day.toLocaleDateString('ru-RU', { day: 'numeric',
month: 'short' }));
  }
  return dates;
};

const handleCellClickInView = (item, dayOfWeek, pairNumber) => {
  // В режиме редактирования всегда можно кликать
  if (item) {
    setEditingItem(item);
    setNewSlot(null);
    setEditModalOpen(true);
  } else {
    setNewSlot({ dayOfWeek, pairNumber });
    setEditingItem(null);
    setEditModalOpen(true);
  }
};

```

```

const handleSaved = async () => {
  if (selectedTeacher) {
    try {
      const [sched, assigns] = await Promise.all([
        getTeacherSchedule(selectedTeacher.id, weekType),
        getTeacherAssignments(selectedTeacher.id)
      ]);
      setTeacherSchedule(sched);
      setTeacherAssignments(assigns);
      setRefreshScheduleKey(prev => prev + 1);
    } catch (err) { console.error(err); }
  }
  fetchPending();
  setEditingItem(null);
  setNewSlot(null);
  setEditModalOpen(false);
};

const handleUpload = async (e) => {
  const file = e.target.files[0];
  if (!file) return;
  const formData = new FormData();
  formData.append('file', file);
  await axiosInstance.post('/api/admin/upload-assignments',
formData);
  alert('Файл загружен');
  fetchPending();
};

const handleExportTeacher = async () => {
  try {
    const response = await axiosInstance.get('/api/admin/export-
schedule', { responseType: 'blob' });
    const url = window.URL.createObjectURL(new
Blob([response.data]));
    const link = document.createElement('a');
    link.href = url;
    link.setAttribute('download', 'schedule_teachers.xlsx');
    document.body.appendChild(link);
    link.click();
    link.remove();
  } catch (error) { console.error(error); }
};

const handleExportStudent = async () => {
  try {
    const response = await axiosInstance.get('/api/admin/export-
student-schedule', { responseType: 'blob' });
    const url = window.URL.createObjectURL(new
Blob([response.data]));
    const link = document.createElement('a');
    link.href = url;
    link.setAttribute('download', 'student_schedule.xlsx');

```

```

        document.body.appendChild(link);
        link.click();
        link.remove();
    } catch (error) { console.error(error); }
};

return (
    <Container maxWidth="xl" sx={{ mt: 4 }}>
        <Typography variant="h4" gutterBottom>Администратор:
{user?.full_name}</Typography>
        <Button variant="outlined" color="error"
onClick={handleLogout} sx={{ mb: 2 }}>Выйти</Button>

        <Box sx={{ mb: 2, display: 'flex', gap: 2 }}>
            <Button variant="contained" color="primary"
onClick={handleExportTeacher}>
                Скачать общую ведомость (Excel)
            </Button>
            <Button variant="contained" color="secondary"
onClick={handleExportStudent}>
                Скачать студенческую ведомость (Excel)
            </Button>
        </Box>

        <Tabs value={tab} onChange={(e, newVal) => setTab(newVal)}
sx={{ mb: 2 }}>
            <Tab label="Управление" />
            <Tab label="Общая ведомость" />
            <Tab label="Студенческая ведомость" />
            <Tab label="Пользователи" />
        </Tabs>

        {tab === 0 && (
            <>
                <Box sx={{ mb: 4 }}>
                    <Typography variant="h6">Загрузить
поручения</Typography>
                    <input type="file" accept=".docx"
onChange={handleUpload} />
                </Box>

                <Typography variant="h6">Расписания на
утверждении</Typography>
                {pendingTeachers.length === 0 ? (
                    <Typography>Нет ожидающих утверждения</Typography>
                ) : (
                    <List>
                        {pendingTeachers.map((teacher) => (
                            <Paper key={teacher.id} sx={{ mb: 1, p: 2 }}>
                                <ListItem>
                                    <ListItemText primary={teacher.full_name}
secondary={teacher.username} />

```

```

        <Button variant="contained" color="info"
onClick={() => handleViewSchedule(teacher)} sx={{ mr:
1 }}>Просмотреть</Button>
        <Button variant="contained" color="secondary"
onClick={() => handleEditSchedule(teacher)} sx={{ mr:
1 }}>Редактировать</Button>
        <Button variant="contained" color="success"
onClick={() => handleApprove(teacher.id)} sx={{ mr:
1 }}>Утвердить</Button>
        <Button variant="contained" color="error"
onClick={() => handleRejectClick(teacher)}>Отклонить</Button>
    </ListItem>
</Paper>
    )})
</List>
)}

    <Dialog open={rejectOpen} onClose={() =>
setRejectOpen(false)}>
        <DialogTitle>Отклонить расписание
{currentRejectTeacher?.full_name}</DialogTitle>
        <DialogContent>
            <TextField label="Причина отклонения" multiline
rows={3} fullWidth value={rejectReason} onChange={(e) =>
setRejectReason(e.target.value)} sx={{ mt: 1 }} />
            <FormControl fullWidth sx={{ mt: 2 }}>
                <InputLabel>Предложить исправленное расписание?
</InputLabel>
                <Select value={proposeEdits ? 'yes' : 'no'}
label="Предложить исправленное расписание?" onChange={(e) =>
setProposeEdits(e.target.value === 'yes')}>
                    <MenuItem value="no">Нет, просто
отклонить</MenuItem>
                    <MenuItem value="yes">Да, предложить
исправленное</MenuItem>
                </Select>
            </FormControl>
        </DialogContent>
        <DialogActions>
            <Button onClick={() =>
setRejectOpen(false)}>Отмена</Button>
            <Button onClick={confirmReject}
color="error">Отклонить</Button>
        </DialogActions>
    </Dialog>
</>
)}

    {tab === 1 && <TeacherScheduleView />}
    {tab === 2 && <StudentScheduleView />}
    {tab === 3 && <UserManagement
onEditSchedule={handleEditUserSchedule} />}

```

```

<Dialog
  open={scheduleOpen}
  onClose={() => setScheduleOpen(false)}
  maxWidth="xl"
  fullWidth
  slotProps={{ paper: { sx: { minHeight: '80vh', minWidth:
'90vw' } } }}
  >
    <DialogTitle>
      Редактирование расписания {selectedTeacher?.full_name}
      <FormControl size="small" sx={{ ml: 2, minWidth: 150 }}>
        <InputLabel>Неделя</InputLabel>
        <Select value={weekOffset} label="Неделя"
onChange={handleWeekChange}>
          {Array.from({ length: 18 }, (_, i) => (
            <MenuItem key={i} value={i}>
              {i + 1}-я ({getDatesForWeek(i)[0]} -
{getDatesForWeek(i)[5]})
            </MenuItem>
          ))}
        </Select>
      </FormControl>
    </DialogTitle>
    <DialogContent sx={{ p: 2 }}>
      <ScheduleGrid
        key={`_${selectedTeacher?.id}-${refreshScheduleKey}`}
        scheduleData={teacherSchedule}
        onCellClick={handleCellClickInView}
        dates={getDatesForWeek(weekOffset)}
      />
    </DialogContent>
  </Dialog>

  {(editingItem || newSlot) && (
    <SlotModal
      open={editModalOpen}
      onClose={() => { setEditModalOpen(false);
setEditingItem(null); setNewSlot(null); }}
      editItem={editingItem}
      dayOfWeek={newSlot?.dayOfWeek}
      pairNumber={newSlot?.pairNumber}
      isAdmin={true}
      assignments={teacherAssignments}
      teacherUserId={selectedTeacher?.id}
      onSave={handleSaved}
      autoApprove={true}
      proposeToTeacher={proposeToTeacher}
    />
  )}
</Container>
);
}

```

Файл «AuthContext.jsx»

```
// src/context/AuthContext.jsx
import { createContext, useContext, useState, useEffect } from
'react';
import axiosInstance from '../api/axiosInstance';

const AuthContext = createContext();

export const useAuth = () => useContext(AuthContext);

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const token = localStorage.getItem('access_token');
    if (token) {
      fetchUser();
    } else {
      setLoading(false);
    }
  }, []);

  const fetchUser = async () => {
    try {
      const response = await axiosInstance.get('/api/auth/me');
      setUser(response.data);
    } catch (error) {
      localStorage.removeItem('access_token');
      setUser(null);
    } finally {
      setLoading(false);
    }
  };

  const login = async (username, password) => {
    const formData = new URLSearchParams();
    formData.append('username', username);
    formData.append('password', password);
    const response = await axiosInstance.post('/api/auth/login',
formData);
    localStorage.setItem('access_token',
response.data.access_token);
    await fetchUser();
  };

  const logout = () => {
    localStorage.removeItem('access_token');
    setUser(null);
  };
};
```

```
    const value = { user, loading, login, logout, isAdmin: user?.role
=== 'admin' };

```

```
    return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
};

```

Файл «UserManagement.jsx»

```
import { useState, useEffect } from 'react';
import {
    Box, Button, Dialog, DialogTitle, DialogContent, DialogActions,
    TextField, FormControl, InputLabel, Select, MenuItem,
    Table, TableBody, TableCell, TableContainer, TableHead, TableRow,
    Paper
} from '@mui/material';
import axiosInstance from '../api/axiosInstance';

```

```
export default function UserManagement({ onEditSchedule }) {
    const [users, setUsers] = useState([]);
    const [open, setOpen] = useState(false);
    const [editUser, setEditUser] = useState(null);
    const [form, setForm] = useState({
        username: '', password: '', full_name: '', role: 'teacher',
        academic_degree: '', department: ''
    });
}

```

```
const fetchUsers = async () => {
    try {
        const res = await axiosInstance.get('/api/admin/users');
        setUsers(res.data);
    } catch (err) { console.error(err); }
};

```

```
useEffect(() => { fetchUsers(); }, []);

```

```
const handleOpenCreate = () => {
    setEditUser(null);
    setForm({ username: '', password: '', full_name: '', role:
'teacher', academic_degree: '', department: '' });
    setOpen(true);
};

```

```
const handleOpenEdit = (user) => {
    setEditUser(user);
    setForm({
        username: user.username,
        password: '',
        full_name: user.full_name,
        role: user.role,
        academic_degree: user.academic_degree || '',
        department: user.department || ''
    });
    setOpen(true);
};

```

```

};

const handleSave = async () => {
  try {
    const payload = { ...form };
    if (!payload.password) delete payload.password;
    if (editUser) {
      await axiosInstance.put(`/api/admin/users/${editUser.id}`,
payload);
    } else {
      await axiosInstance.post(`/api/admin/users`, payload);
    }
    setOpen(false);
    fetchUsers();
  } catch (err) {
    alert(err.response?.data?.detail || 'Ошибка сохранения');
  }
};

const handleDelete = async (user) => {
  if (user.role === 'admin') return; // дополнительная защита
  if (!window.confirm(`Удалить пользователя ${user.full_name}?`))
return;
  try {
    await axiosInstance.delete(`/api/admin/users/${user.id}`);
    fetchUsers();
  } catch (err) {
    alert(err.response?.data?.detail || 'Ошибка удаления');
  }
};

return (
  <Box>
    <Button variant="contained" onClick={handleOpenCreate}
sx={{ mb: 2 }}>Добавить пользователя</Button>
    <TableContainer component={Paper}>
      <Table size="small">
        <TableHead>
          <TableRow>
            <TableCell>ID</TableCell>
            <TableCell>Логин</TableCell>
            <TableCell>ФИО</TableCell>
            <TableCell>Роль</TableCell>
            <TableCell>Учёная степень</TableCell>
            <TableCell>Кафедра</TableCell>
            <TableCell>Действия</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {users.map((u) => (
            <TableRow key={u.id}>
              <TableCell>{u.id}</TableCell>
              <TableCell>{u.username}</TableCell>

```

```

        <TableCell>{u.full_name}</TableCell>
        <TableCell>{u.role}</TableCell>
        <TableCell>{u.academic_degree || ''}</TableCell>
        <TableCell>{u.department || ''}</TableCell>
        <TableCell>
            <Button size="small" variant="outlined"
onClick={ () => handleOpenEdit(u)} sx={{ mr: 1 }}>Изменить</Button>
            {u.role === 'teacher' && (
                <Button size="small" variant="outlined"
color="secondary" onClick={ () => onEditSchedule &&
onEditSchedule(u)} sx={{ mr: 1 }}>
                Расписание
            </Button>
            )}
            {u.role !== 'admin' && (
                <Button size="small" variant="outlined"
color="error" onClick={ () => handleDelete(u)}>Удалить</Button>
            )}
        </TableCell>
    </TableRow>
    )}
</TableBody>
</Table>
</TableContainer>

    <Dialog open={open} onClose={ () => setOpen(false)}
maxWidth="sm" fullWidth>
        <DialogTitle>{editUser ? 'Редактировать пользователя' :
'Создать пользователя'}</DialogTitle>
        <DialogContent>
            <TextField label="Логин" fullWidth margin="normal"
value={form.username}
            onChange={ (e) => setForm({ ...form, username:
e.target.value })} />
            <TextField label="Пароль (оставьте пустым, чтобы не
менять)" fullWidth margin="normal" type="password"
            value={form.password} onChange={ (e) => setForm({
...form, password: e.target.value })} />
            <TextField label="ФИО" fullWidth margin="normal"
value={form.full_name}
            onChange={ (e) => setForm({ ...form, full_name:
e.target.value })} />
            <FormControl fullWidth margin="normal">
                <InputLabel>Роль</InputLabel>
                <Select value={form.role} label="Роль" onChange={ (e) =>
setForm({ ...form, role: e.target.value })}>
                    <MenuItem value="teacher">Преподаватель</MenuItem>
                    <MenuItem value="admin">Администратор</MenuItem>
                </Select>
            </FormControl>
            <TextField label="Учёная степень" fullWidth
margin="normal" value={form.academic_degree}

```

```

        onChange={ (e) => setForm({ ...form, academic_degree:
e.target.value })} />
        <TextField label="Кафедра" fullWidth margin="normal"
value={form.department}
        onChange={ (e) => setForm({ ...form, department:
e.target.value })} />
      </DialogContent>
      <DialogActions>
        <Button onClick={() => setOpen(false)}>Отмена</Button>
        <Button onClick={handleSave}
variant="contained">Сохранить</Button>
      </DialogActions>
    </Dialog>
  </Box>
);
}

```

Файл «TeacherScheduleView.jsx»

```

import { useState, useEffect } from 'react';
import { Box, CircularProgress, Typography } from '@mui/material';
import { getPublicScheduleData, getMyScheduleData } from
'../api/teacherApi';

export default function TeacherScheduleView({ personal = false }) {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchData = personal ? getMyScheduleData :
getPublicScheduleData;
    fetchData().then(setData).catch(console.error).finally(() =>
setLoading(false));
  }, [personal]);

  if (loading) return <Box sx={{ display: 'flex', justifyContent:
'center', mt: 4 }}><CircularProgress /></Box>;
  if (!data) return <Typography color="error">Не удалось загрузить
данные</Typography>;

  const { days, time_slots, rows } = data;

  return (
    <Box sx={{ overflowX: 'auto', mt: 2 }}>
      <table style={{ borderCollapse: 'collapse', width: '100%',
fontSize: '0.85rem' }}>
        <thead>
          <tr>
            <th style={{ border: '1px solid #ccc', padding: 4,
background: '#e0e0e0', width: 30 }}>№</th>
            <th style={{ border: '1px solid #ccc', padding: 4,
background: '#e0e0e0', width: 120 }}>Ф.И.О.</th>
            {days.map((day, idx) => (

```

```

        <th key={idx} colSpan={time_slots.length}
style={{ border: '2px solid #666', padding: 4, background:
'#e0e0e0', textAlign: 'center' }}>
            {day}
        </th>
    )})
</tr>
<tr>
    <th style={{ border: '1px solid #ccc', background:
'#e0e0e0' }}></th>
    <th style={{ border: '1px solid #ccc', background:
'#e0e0e0' }}></th>
    {days.map(() =>
        time_slots.map((slot, i) => (
            <th key={i} style={{ border: '1px solid #ccc',
padding: 2, background: '#e0e0e0', fontSize: '0.7rem', fontWeight:
'normal', textAlign: 'center', whiteSpace: 'nowrap' }}>
                {slot}
            </th>
        ))
    )}
</tr>
</thead>
<tbody>
    {rows.map((row) => (
        <tr key={row.num}>
            <td style={{ border: '1px solid #ccc', padding: 4,
textAlign: 'center' }}>{row.num}</td>
            <td style={{ border: '1px solid #ccc', padding: 4,
whiteSpace: 'nowrap' }}>{row.teacher_name}</td>
            {days.map((_, dayIdx) =>
                time_slots.map((_, pairIdx) => {
                    const key = `${dayIdx}-${pairIdx}`;
                    const val = row.cells[key] || '';
                    return (
                        <td key={key} style={{ border: '1px solid #ccc',
padding: 4, fontSize: '0.75rem', verticalAlign: 'top', whiteSpace:
'pre-wrap' }}>
                            {val}
                        </td>
                    );
                })
            )}
        </tr>
    )})
</tbody>
</table>
</Box>
);
}

```

Файл «StudentScheduleView.jsx»

```
import { useState, useEffect } from 'react';
```

```

import { getStudentScheduleData } from '../api/adminApi';
import { Box, CircularProgress, Typography } from '@mui/material';
import React from 'react'
export default function StudentScheduleView() {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {

getStudentScheduleData().then(setData).catch(console.error).finally(
  () => setLoading(false));
  }, []);

  if (loading) return <Box sx={{ display: 'flex', justifyContent:
'center', mt: 4 }}><CircularProgress /></Box>;
  if (!data) return <Typography color="error">Не удалось загрузить
данные</Typography>;

  const { days_data, groups } = data;

  return (
    <Box sx={{ overflowX: 'auto', mt: 2, fontSize: '0.8rem' }}>
      <table style={{ borderCollapse: 'collapse', width: '100%',
minWidth: 800 }}>
        <thead>
          <tr style={{ background: '#e0e0e0' }}>
            <th style={{ border: '1px solid #ccc', padding: 4,
width: 40 }}>дни</th>
            <th style={{ border: '1px solid #ccc', padding: 4,
width: 80 }}>часы</th>
            {groups.map((g, i) => (
              <th key={i} style={{ border: '2px solid #666',
padding: 4, textAlign: 'center' }}>
                {g.name}
              </th>
            ))}
          </tr>
        </thead>
        <tbody>
          {days_data.map((dayObj) =>
            dayObj.slots.map((slot, slotIdx) => {
              const hasSplit = slot.cells.some(c => c.odd_text ||
c.even_text);

              return (
                <React.Fragment key={`_${dayObj.day}-${slot.time}`}>
                  <tr>
                    {slotIdx === 0 && (
                      <td rowSpan={dayObj.slots.length}
style={{ border: '1px solid #ccc', textAlign: 'center', fontWeight:
'bold', verticalAlign: 'middle' }}>
                        {dayObj.day}
                      </td>

```

```

    ))
    <td style={{ border: '1px solid #ccc',
textAlign: 'center', whiteSpace: 'nowrap' }}>{slot.time}</td>
    {slot.cells.map((cell, gIdx) => (
      <td key={gIdx} style={{ border: '1px solid
#ccc', padding: 4, verticalAlign: 'top', whiteSpace: 'pre-wrap' }}>
        {cell.all_text ? cell.all_text :
(cell.odd_text || '')}
      </td>
    ))}
  </tr>
  {hasSplit && (
    <tr>
      {slotIdx === 0 && <td style={{ display: 'none'
}} />}

      <td style={{ border: '1px solid #ccc',
textAlign: 'center', whiteSpace: 'nowrap' }}>{slot.time}</td>
      {slot.cells.map((cell, gIdx) => (
        <td key={gIdx} style={{ border: '1px solid
#ccc', padding: 4, verticalAlign: 'top', whiteSpace: 'pre-wrap' }}>
          {cell.even_text || ''}
        </td>
      ))}
    </tr>
  )}
</React.Fragment>
);
  })
})
</tbody>
</table>
</Box>
);
}

```

Файл «SlotModal.jsx»

```

import {
  Dialog, DialogTitle, DialogContent, FormControl, InputLabel,
  Select, MenuItem, Button, Alert, Box, TextField, DialogActions,
  Typography, Stack
} from '@mui/material';
import { useState, useEffect } from 'react';
import {
  getMyAssignments, getAvailableSlots, createScheduleItem,
  updateScheduleItem, deleteScheduleItem,
  acceptCollaboration, rejectCollaboration
} from '../api/teacherApi';
import {
  adminUpdateScheduleItem, adminDeleteScheduleItem,
  adminCreateScheduleItem, adminGetAvailableSlots
} from '../api/adminApi';

```

```

export default function SlotModal({ open, onClose, dayOfWeek,
pairNumber, onSave, editItem, isAdmin, assignments, teacherUserId,
proposeToTeacher }) {
  const isEdit = Boolean(editItem);
  // Приглашение - если invited_user_id определён (не важно, draft
или collaboration_waiting)
  const isCollaborationInvite = !isAdmin &&
editItem?.invited_user_id != null;
  const [allAssignments, setAllAssignments] = useState([]);
  const [selectedAssignment, setSelectedAssignment] = useState('');
  const [classrooms, setClassrooms] = useState([]);
  const [selectedClassroom, setSelectedClassroom] = useState('');
  const [weekType, setWeekType] = useState('all');
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const [rejectOpen, setRejectOpen] = useState(false);
  const [rejectReason, setRejectReason] = useState('');

  useEffect(() => {
    if (!open) return;

    // Всегда загружаем список назначений (для избежания ошибки out-
of-range)
    if (assignments) {
      setAllAssignments(assignments);
    } else {

getMyAssignments().then(setAllAssignments).catch(console.error);
    }

    // Загружаем данные для текущего занятия
    if (editItem) {
      setSelectedAssignment(editItem.teaching_assignment_id);
      setWeekType(editItem.week_type || 'all');
      const d = dayOfWeek ?? editItem.day_of_week;
      const p = pairNumber ?? editItem.pair_number;
      const fetchSlots = isAdmin ? adminGetAvailableSlots :
getAvailableSlots;
      fetchSlots(editItem.teaching_assignment_id, d, p)
        .then(data => {
          let rooms = data.free_classrooms || [];
          if (editItem.classroom_id && !rooms.some(r => r.id ===
editItem.classroom_id)) {
            rooms.push({
              id: editItem.classroom_id,
              number: editItem.classroom_number || `Ауд. $
{editItem.classroom_id}`,
              has_projector: false,
              has_computers: false,
            });
          }
          setClassrooms(rooms);
          setSelectedClassroom(editItem.classroom_id || '');

```

```

    })
    .catch(err => {
      console.error(err);
      setError('Не удалось загрузить аудитории');
    });
  } else {
    setSelectedAssignment('');
    setSelectedClassroom('');
    setWeekType('all');
    setClassrooms([]);
  }
}, [open, editItem, assignments, isAdmin]);

const handleAssignmentChange = async (e) => {
  const id = e.target.value;
  setSelectedAssignment(id);
  setSelectedClassroom('');
  setError('');
  try {
    const d = dayOfWeek ?? editItem?.day_of_week;
    const p = pairNumber ?? editItem?.pair_number;
    const fetchSlots = isAdmin ? adminGetAvailableSlots :
getAvailableSlots;
    const data = await fetchSlots(id, d, p);
    setClassrooms(data.free_classrooms || []);
    if (!data.slot_free) setError('Этот слот занят для выбранного
назначения');
  } catch (err) {
    console.error(err);
    setError('Ошибка загрузки аудиторий');
  }
};

const handleSave = async () => {
  if (isCollaborationInvite) return;
  setError('');
  setLoading(true);
  try {
    const payload = {
      teaching_assignment_id: parseInt(selectedAssignment),
      classroom_id: parseInt(selectedClassroom),
      day_of_week: dayOfWeek ?? editItem.day_of_week,
      pair_number: pairNumber ?? editItem.pair_number,
      week_type: weekType,
    };
    if (isEdit) {
      if (isAdmin) {
        const params = proposeToTeacher ? { propose: true } : {};
        await adminUpdateScheduleItem(editItem.id, payload,
params);
      } else {
        await updateScheduleItem(editItem.id, payload);
      }
    }
  }

```

```

    } else {
      if (isAdmin && teacherUserId) {
        const params = proposeToTeacher ? { propose: true } : {};
        await adminCreateScheduleItem(teacherUserId, payload,
params);
      } else {
        await createScheduleItem(payload);
      }
    }
    onSave();
    onClose();
  } catch (err) {
    const msg = err.response?.data?.detail;
    setError(typeof msg === 'string' ? msg : JSON.stringify(msg));
  } finally { setLoading(false); }
};

```

```

const handleDelete = async () => {
  if (isCollaborationInvite) return;
  if (!editItem) return;
  setLoading(true);
  try {
    if (isAdmin) {
      await adminDeleteScheduleItem(editItem.id);
    } else {
      await deleteScheduleItem(editItem.id);
    }
    onSave();
    onClose();
  } catch (err) {
    const msg = err.response?.data?.detail;
    setError(typeof msg === 'string' ? msg : JSON.stringify(msg));
  } finally { setLoading(false); }
};

```

```

const handleAccept = async () => {
  setLoading(true);
  try {
    await acceptCollaboration(editItem.id);
    onSave();
    onClose();
  } catch (err) {
    const msg = err.response?.data?.detail;
    setError(typeof msg === 'string' ? msg : JSON.stringify(msg));
  } finally { setLoading(false); }
};

```

```

const handleRejectOpen = () => {
  setRejectOpen(true);
};

```

```

const handleRejectConfirm = async () => {
  setLoading(true);

```

```

    try {
      await rejectCollaboration(editItem.id, rejectReason);
      setRejectOpen(false);
      onSave();
      onClose();
    } catch (err) {
      const msg = err.response?.data?.detail;
      setError(typeof msg === 'string' ? msg : JSON.stringify(msg));
    } finally { setLoading(false); }
  };

  const titleDay = dayOfWeek ? ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб']
[dayOfWeek-1] : '';
  const titlePair = pairNumber ?? '';
  const title = isEdit
    ? `Редактирование: ${editItem.discipline_name || ''}`
    : `Добавить занятие: ${titleDay}, ${titlePair}-я пара`;

  return (
    <>
    <Dialog open={open} onClose={onClose} fullWidth maxWidth="sm">
      <DialogTitle>{title}</DialogTitle>
      <DialogContent>
        {error && <Alert severity="error" sx={{ mb:
2 }}>{error}</Alert>}

        {/* Режим отображения для приглашённого преподавателя */}
        {isCollaborationInvite && (
          <Stack spacing={2} sx={{ mb: 2 }}>
            <Box>
              <Typography variant="body2"
color="text.secondary">Дисциплина / группа</Typography>
              <Typography
variant="body1">{editItem.discipline_name}
({editItem.group_name})</Typography>
            </Box>
            <Box>
              <Typography variant="body2"
color="text.secondary">Аудитория</Typography>
              <Typography
variant="body1">{editItem.classroom_number || '-'}</Typography>
            </Box>
            <Box>
              <Typography variant="body2"
color="text.secondary">Тип недели</Typography>
              <Typography variant="body1">
                {editItem.week_type === 'all' ? 'Каждую неделю' :
editItem.week_type === 'odd' ? 'Только по нечётным' : 'Только по
чётным'}
              </Typography>
            </Box>
          <Box sx={{ display: 'flex', gap: 2 }}>

```

```

        <Button variant="contained" color="success"
onClick={handleAccept} disabled={loading}>
        Принять
    </Button>
    <Button variant="outlined" color="error"
onClick={handleRejectOpen} disabled={loading}>
        Отклонить
    </Button>
</Box>
</Stack>
    )}

    {/* Обычный режим редактирования (не для приглашённого)
*/}

    {!isCollaborationInvite && (
        <>
            <FormControl fullWidth sx={{ mt: 1 }}>
                <InputLabel>Дисциплина / группа</InputLabel>
                <Select value={selectedAssignment} label="Дисциплина
/ группа" onChange={handleAssignmentChange}>
                    {allAssignments.map((a) => (
                        <MenuItem key={a.id} value={a.id}>
                            {a.discipline_name} ({a.group_name},
{a.lesson_type})
                        </MenuItem>
                    ))}
                </Select>
            </FormControl>

            <FormControl fullWidth sx={{ mt: 2 }}>
                <InputLabel>Аудитория</InputLabel>
                <Select
                    value={selectedClassroom}
                    label="Аудитория"
                    onChange={(e) =>
setSelectedClassroom(e.target.value)}
                >
                    {classrooms.length > 0 ? (
                        classrooms.map((c) => (
                            <MenuItem key={c.id} value={c.id}>
                                {c.number}
                                {c.has_projector && ' есть проектор, '}
                                {c.has_computers && ' есть компьютер '}
                                {!c.suitable && ' (не подходит)'}
                            </MenuItem>
                        ))
                    ) : (
                        <MenuItem disabled value="">
                            <em>Нет доступных аудиторий</em>
                        </MenuItem>
                    )}
                </Select>
            </FormControl>

```

```

        <FormControl fullWidth sx={{ mt: 2 }}>
          <InputLabel>Тип недели</InputLabel>
          <Select value={weekType} label="Тип недели"
onChange={ (e) => setWeekType(e.target.value)}>
            <MenuItem value="all">Каждую неделю</MenuItem>
            <MenuItem value="odd">Только по
нечётным</MenuItem>
            <MenuItem value="even">Только по чётным</MenuItem>
          </Select>
        </FormControl>

        <Box sx={{ display: 'flex', gap: 2, mt: 3 }}>
          <Button variant="contained" fullWidth disabled={!
selectedAssignment || !selectedClassroom || loading}
onClick={handleSave}>
            {isEdit ? 'Сохранить изменения' : 'Создать'}
          </Button>
          {isEdit && (
            <Button variant="outlined" color="error" fullWidth
onClick={handleDelete} disabled={loading}>
              Удалить
            </Button>
          )}
        </Box>
      </>
    )}
  </DialogContent>
</Dialog>

  <Dialog open={rejectOpen} onClose={() =>
setRejectOpen(false)}>
    <DialogTitle>Отклонить предложение</DialogTitle>
    <DialogContent>
      <TextField
        label="Причина отклонения"
        multiline
        rows={3}
        fullWidth
        value={rejectReason}
        onChange={(e) => setRejectReason(e.target.value)}
        sx={{ mt: 1 }}
      />
    </DialogContent>
    <DialogActions>
      <Button onClick={() =>
setRejectOpen(false)}>Отмена</Button>
      <Button onClick={handleRejectConfirm}
color="error">Отклонить</Button>
    </DialogActions>
  </Dialog>
</>
);

```

```
}
```

Файл «ScheduleGrid.jsx»

```
import { useState, useEffect } from 'react';
import { Box, Typography, Paper } from '@mui/material';
import { getMySchedule } from '../api/teacherApi';

const DAYS = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб'];
const PAIRS = [1, 2, 3, 4, 5, 6];

export default function ScheduleGrid({ onCellClick, onDraftClick,
refreshKey, dates, weekType, scheduleData }) {
  const [schedule, setSchedule] = useState([]);

  useEffect(() => {
    if (scheduleData !== undefined) {
      setSchedule(scheduleData);
    } else {

getMySchedule(weekType).then(setSchedule).catch(console.error);
    }
    }, [refreshKey, weekType, scheduleData]);

    const getCellData = (day, pair) => {
      return schedule.find(
        (item) => item.day_of_week === day && item.pair_number ===
pair
      );
    };

    const getCellColor = (item) => {
      if (!item) return '#e8f5e9'; // свободно
      if (item.status === 'collaboration_waiting') return
'#ffe0b2'; // оранжевый
      if (item.proposed_by_admin) return '#f3e5f5'; // предложено
админом
      if (item.status === 'draft') return '#bbdefb';
      if (item.status === 'pending') return '#fff9c4';
      if (item.status === 'approved') return '#ffcdd2';
      return '#e0e0e0';
    };

    const handleCellClickInternal = (item, dayNum, pair) => {
      if (scheduleData !== undefined) {
        onCellClick?.(item, dayNum, pair);
        return;
      }
      if (!item) {
        onCellClick?.(dayNum, pair);
      } else if (item.status === 'draft' || item.status ===
'collaboration_invited') {
        onDraftClick?.(item);
      }
    }
  }
}
```

```

};

return (
  <Box>
    <Box
      sx={{
        display: 'grid',
        gridTemplateColumns: '50px repeat(6, 1fr)',
        gap: '4px',
        width: '100%',
        overflowX: 'auto',
      }}
    >
      <Box />
      {DAYS.map((day, idx) => (
        <Box
          key={day}
          sx={{
            textAlign: 'center',
            fontWeight: 'bold',
            p: 1,
            backgroundColor: '#f5f5f5',
            borderRadius: 1,
          }}
        >
          {day}
          {dates && (
            <Typography variant="caption" display="block">
              {dates[idx]}
            </Typography>
          )}
        </Box>
      ))}

      {PAIRS.map((pair) => (
        <Box key={pair} sx={{ display: 'contents' }}>
          <Paper
            sx={{
              display: 'flex',
              alignItems: 'center',
              justifyContent: 'center',
              fontWeight: 'bold',
              fontSize: 18,
              p: 2,
            }}
          >
            {pair}
          </Paper>

          {DAYS.map((_, idx) => {
            const dayNum = idx + 1;
            const item = getCellData(dayNum, pair);
            const backgroundColor = getCellColor(item);

```

```

return (
  <Paper
    key={`${dayNum}-${pair}`}
    sx={{
      minHeight: 120,
      p: 1,
      display: 'flex',
      flexDirection: 'column',
      justifyContent: 'center',
      backgroundColor,
      border: '1px solid #ccc',
      cursor: (!item || item.status === 'draft' ||
item.status === 'collaboration_invited' || (scheduleData !==
undefined && item?.status === 'pending')) ? 'pointer' : 'default',
      overflow: 'hidden',
      wordBreak: 'break-word',
    }}
    onClick={() => handleCellClickInternal(item,
dayNum, pair)}
  >
    {item ? (
      <>
        <Typography variant="body2" fontWeight="bold">
          {item.discipline_name}
        </Typography>
        <Typography variant="caption">
          {item.group_name} ({item.lesson_type ===
'lecture' ? 'лекция' : 'лаб.'})
        </Typography>
        <Typography variant="caption">
          {item.teacher_name}
        </Typography>
        <Typography variant="caption"
color="text.secondary">
          ауд. {item.classroom_number}
        </Typography>
      </>
    ) : (
      <Typography variant="caption"
color="text.secondary" sx={{ textAlign: 'center' }}>
        Свободно
      </Typography>
    )}
  </Paper>
);
}}
</Box>
)}}
</Box>

{/* Легенда */}
<Box sx={{ mt: 2, display: 'flex', flexWrap: 'wrap', gap: 1.5,
fontSize: '0.75rem', color: 'text.secondary' }}>

```

```

    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#e8f5e9', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>Свободно</span>
    </Box>
    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#bbdefb', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>Черновик</span>
    </Box>
    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#fff9c4', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>На утверждении</span>
    </Box>
    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#ffcdd2', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>Утверждено</span>
    </Box>
    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#f3e5f5', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>Предложено адм.</span>
    </Box>
    <Box sx={{ display: 'flex', alignItems: 'center', gap:
0.5 }}>
      <Box sx={{ width: 12, height: 12, backgroundColor:
'#ffe0b2', border: '1px solid #ccc', flexShrink: 0 }} />
      <span>Приглашение</span>
    </Box>
  </Box>
);
}

```

Файл «teacherApi.js»

```

import axiosInstance from './axiosInstance';
import TeacherScheduleView from '../components/TeacherScheduleView';
import StudentScheduleView from '../components/StudentScheduleView';
export const getMyAssignments = async () => {
  const response = await axiosInstance.get('/api/teacher/my-
assignments');
  return response.data;
};

export const getMySchedule = async (weekType = null) => {
  const params = {};

```

```

    if (weekType) params.week_type = weekType;
    const response = await axiosInstance.get('/api/teacher/my-schedule', { params });
    return response.data;
  };

export const getAvailableSlots = async (teachingAssignmentId, dayOfWeek, pairNumber) => {
  const response = await axiosInstance.get('/api/teacher/available-slots', {
    params: { teaching_assignment_id: teachingAssignmentId, day_of_week: dayOfWeek, pair_number: pairNumber },
  });
  return response.data;
};

export const createScheduleItem = async (data) => {
  const response = await axiosInstance.post('/api/teacher/schedule-items', data);
  return response.data;
};

export const updateScheduleItem = async (id, data) => {
  const response = await axiosInstance.put(`/api/teacher/schedule-items/${id}`, data);
  return response.data;
};

export const deleteScheduleItem = async (id) => {
  await axiosInstance.delete(`/api/teacher/schedule-items/${id}`);
};

export const submitSchedule = async (note = null) => {
  const params = note ? { note } : {};
  const response = await axiosInstance.post('/api/teacher/schedule-items/submit', null, { params });
  return response.data;
};

export const clearProposed = async () => {
  await axiosInstance.post('/api/teacher/clear-proposed');
};

export const getMyScheduleData = async () => {
  const response = await axiosInstance.get('/api/teacher/my-schedule-data');
  return response.data;
};

export const getPublicScheduleData = async () => {
  const response = await axiosInstance.get('/api/teacher/public-schedule-data');
  return response.data;
};

```

```

};

export const acceptCollaboration = async (itemId, note = null) => {
  const params = note ? { note } : {};
  const response = await axiosInstance.post(`/api/teacher/schedule-items/${itemId}/accept-collaboration`, null, { params });
  return response.data;
};

export const rejectCollaboration = async (itemId, reason = null) => {
  const params = reason ? { reason } : {};
  const response = await axiosInstance.post(`/api/teacher/schedule-items/${itemId}/reject-collaboration`, null, { params });
  return response.data;
};

```

Файл «axiosInstance.js»

```

import axios from 'axios';

const axiosInstance = axios.create({
  baseURL: 'http://127.0.0.1:8000',
});

// Добавляем токен к каждому запросу, если он есть
axiosInstance.interceptors.request.use((config) => {
  const token = localStorage.getItem('access_token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

// Обработка 401: редирект на логин при истечении токена
axiosInstance.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      localStorage.removeItem('access_token');
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);

```

export default axiosInstance;

Файл «adminApi.js»

```

import axiosInstance from './axiosInstance';

export const getTeacherSchedule = async (teacherUserId, weekType) => {
  const params = {};

```

```

    if (weekType) params.week_type = weekType;
    const response = await axiosInstance.get(`/api/admin/schedule/${teacherUserId}`, { params });
    return response.data;
  };

export const adminUpdateScheduleItem = async (itemId, data, params = {}) => {
  const response = await axiosInstance.put(`/api/admin/schedule-items/${itemId}`, data, { params });
  return response.data;
};

export const adminDeleteScheduleItem = async (itemId) => {
  await axiosInstance.delete(`/api/admin/schedule-items/${itemId}`);
};

export const getTeacherAssignments = async (teacherUserId) => {
  const response = await axiosInstance.get(`/api/admin/teacher/${teacherUserId}/assignments`);
  return response.data;
};

export const adminCreateScheduleItem = async (teacherUserId, data, params = {}) => {
  const response = await axiosInstance.post(`/api/admin/schedule-items?teacher_user_id=${teacherUserId}`, data, { params });
  return response.data;
};

export const adminGetAvailableSlots = async (teachingAssignmentId, dayOfWeek, pairNumber) => {
  const response = await axiosInstance.get('/api/admin/available-slots', {
    params: {
      teaching_assignment_id: teachingAssignmentId,
      day_of_week: dayOfWeek,
      pair_number: pairNumber,
    },
  });
  return response.data;
};

export const getScheduleData = async () => {
  const response = await axiosInstance.get('/api/admin/schedule-data');
  return response.data;
};

export const getStudentScheduleData = async () => {
  const response = await axiosInstance.get('/api/admin/student-schedule-data');
  return response.data;
};

```

};

## **Приложение В**

**(обязательное)**

Web-приложение составления расписания ВУЗа

Презентация

# Web-приложение составления расписания ВУЗа

Автор: Курушин Ярослав Сергеевич, гр. 22BBB1

Руководитель: д.т.н., профессор Зинкин С.А.

## Цель и задачи

**Цель** – разработать web-приложение для автоматизированного составления семестрового расписания вуза на основе файла поручений Microsoft Word.

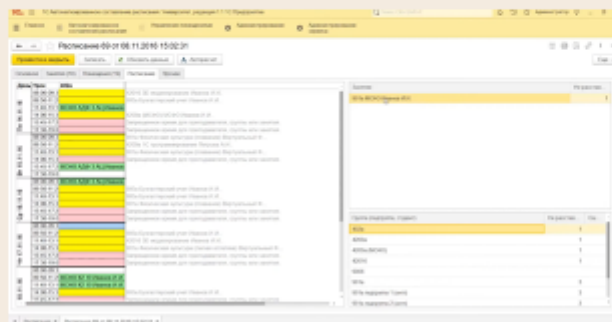
**Задачи:**

- Проанализировать предметную область и аналоги.
- Спроектировать архитектуру и базу данных.
- Реализовать парсер .docx, серверную логику (FastAPI), клиент (React).
- Обеспечить ролевой доступ (администратор / преподаватель).
- Добавить валидацию конфликтов и механизм совместных занятий.
- Реализовать экспорт расписания в Excel.

## Аналог – «1С:Автоматизированное составление расписаний. Университет»

**Достоинства:** Интеграция, мощные алгоритмы

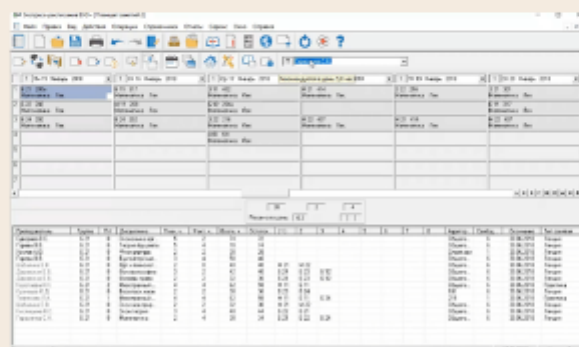
**Недостатки:** Дорого, централизованно, нет импорта из Word-поручений



## Аналог – «Экспресс-расписание ВУЗ»

• **Достоинства:** Быстрая генерация, удобный интерфейс

• **Недостатки :** Только Windows, однопользовательская, десктопная



## Вывод по аналогам

- Ни одно из рассмотренных решений не удовлетворяет требованиям децентрализации и импорта из .docx.
- Целесообразна разработка собственного специализированного Web-приложения.

## Стек технологий

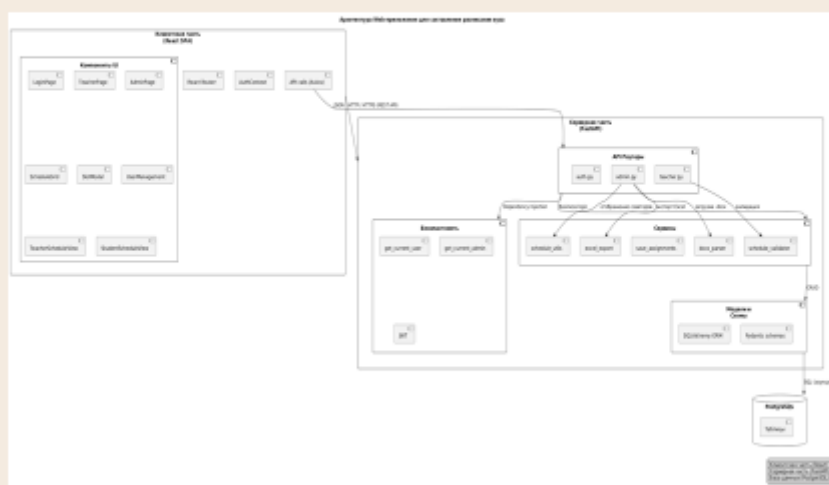
- Backend: Python, FastAPI
- Frontend: JavaScript, React
- База данных: PostgreSQL



## Требования к программе

- Импорт поручений из .docx.
- Роли: администратор / преподаватель.
- CRUD занятий, групп, аудиторий, дисциплин.
- Автоматическая валидация: занятость аудитории, группы, преподавателя; лимит 4 пары в день; соответствие оборудования (проектор, компьютеры).
- Совместные лабораторные занятия (приглашения, разделение групп).
- Отправка на утверждение → утверждение/отклонение администратором.
- Экспорт в Excel (преподавательская и студенческая ведомости).

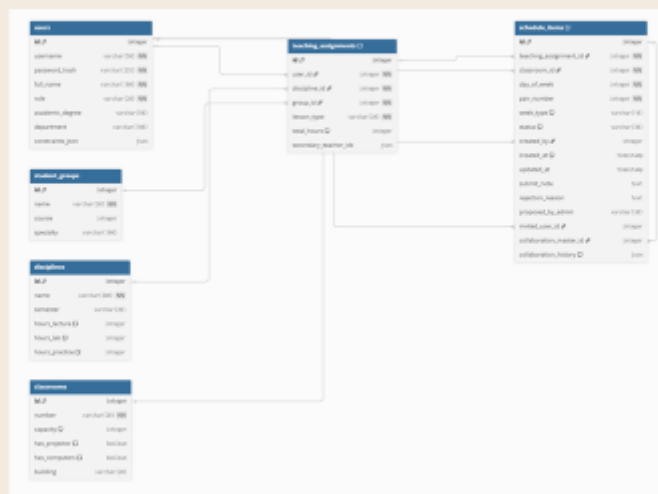
## Архитектура приложения



### Диаграмма вариантов использования



## Модель базы данных



## Пользовательский интерфейс – Преподаватель

Добро пожаловать, Акифьев И.В.!

Администрирование | Личная информация | **История оценок** | Прогнозирование успеваемости | Группы | Тестирование (20 вопросов) | 10 вопросов (20 вопросов) | 10 вопросов (20 вопросов)

| Вопрос                                                                                                                                               | Вопрос | Вопрос | Вопрос | Вопрос | Вопрос |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|--------|--------|--------|
| 1. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |
| 2. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |
| 3. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |
| 4. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |
| 5. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |
| 6. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  | Верно  | Верно  |

Справка | Настройки | Статистика | История | Прогнозирование | Тестирование

## Пользовательский интерфейс – Преподаватель

Добавить задание: ИТ, 5-я пара

Дисциплина / группа:

Аудитория:

Тип задания:

Кандидат ответа:

Добро пожаловать, Дубинин В.И.!

Администрирование | Личная информация | **История оценок** | Прогнозирование успеваемости | Группы | Тестирование (20 вопросов) | 10 вопросов (20 вопросов) | 10 вопросов (20 вопросов)

| Вопрос                                                                                                                                               | Вопрос | Вопрос | Вопрос |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|--------|
| 1. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  |
| 2. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  | Верно  |

Добро пожаловать, Акифьев И.В.!

Администрирование | Личная информация | **История оценок** | Прогнозирование успеваемости | Группы | Тестирование (20 вопросов) | 10 вопросов (20 вопросов) | 10 вопросов (20 вопросов)

| Вопрос                                                                                                                                               | Вопрос | Вопрос |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|
| 1. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  |
| 2. Декларативная модель программирования (DMM) – это модель, в которой программа описывается декларативно, без указания порядка выполнения операций. | Верно  | Верно  |

Редистрирование: Декларативные языки программирования

Дисциплина / группа:

Декларативные языки программирования (DMM)

Аудитория:

Тип задания:

Кандидат ответа:

## Пользовательский интерфейс – Преподаватель

| УЧАСТУЮЩИЕ |                  | СТУДЕНЧЕСКАЯ ДЕЯТЕЛЬНОСТЬ |         |                                                                    |          |          |          |          |          |         |          |          |          | ПОЛЬЗОВАТЕЛИ |                                                                    |  |
|------------|------------------|---------------------------|---------|--------------------------------------------------------------------|----------|----------|----------|----------|----------|---------|----------|----------|----------|--------------|--------------------------------------------------------------------|--|
| №          | Ф.И.О.           | КУРСОВЫЕ ПРОЕКТЫ          |         |                                                                    |          |          |          |          |          |         |          |          |          | ИТОГО        |                                                                    |  |
|            |                  | 10.04.19                  | 9.05.19 | 11.06.19                                                           | 10.07.19 | 10.08.19 | 17.09.19 | 10.10.19 | 10.11.19 | 9.12.19 | 10.01.20 | 10.02.20 | 10.03.20 | 10.04.20     | 10.05.20                                                           |  |
| 1          | Дубовик В.Н.     |                           |         | 1400011<br>Дипломное<br>исс<br>по: Дубовик В.Н.<br>от 14.10 (100%) |          |          |          |          |          |         |          |          |          |              | 1400011<br>Дипломное<br>исс<br>по: Дубовик В.Н.<br>от 14.10 (100%) |  |
| 2          | Андреев Е.В.     |                           |         | 1400012<br>Дипломное<br>исс<br>по: Андреев Е.В.<br>от 14.10        |          |          |          |          |          |         |          |          |          |              | 1400012<br>Дипломное<br>исс<br>по: Андреев Е.В.<br>от 14.10        |  |
| 3          | Карпенченко Н.С. |                           |         | 120001<br>Дипломное<br>исс<br>по: Карпенченко<br>Н.С. от 14.10     |          |          |          |          |          |         |          |          |          |              | 120001<br>Дипломное<br>исс<br>по: Карпенченко<br>Н.С. от 14.10     |  |
| 4          | Грица Е.Е.       |                           |         |                                                                    |          |          |          |          |          |         |          |          |          |              | 120001<br>Дипломное<br>исс<br>по: Грица<br>Е.Е. от 14.10           |  |
| 4          |                  |                           |         |                                                                    |          |          |          |          |          |         |          |          |          |              |                                                                    |  |

## Пользовательский интерфейс – Администратор

Адаптировано IntelliJ

Узнать больше | Скачать | Лицензия | Помощь | Блог | Связаться с нами

Получите IntelliJ IDEA бесплатно

Почему IntelliJ IDEA

Скачайте IntelliJ IDEA

Отложить расписание Акафиев Н.В.

Причины отклонения

Продолжить рассмотрение расписания?

Да, продолжить рассмотрение

ОТМЕНА ОТКАЗАТЬ

Администрация ЗМЕТТ

Административный персонал администрации

| № п/п | Имя                         | Должность             | Служба | Служба | Служба | Служба |
|-------|-----------------------------|-----------------------|--------|--------|--------|--------|
| 1     | Иванов Иван Иванович        | Директор              | Служба | Служба | Служба | Служба |
| 2     | Петров Петр Петрович        | Заместитель директора | Служба | Служба | Служба | Служба |
| 3     | Сидоров Сергей Сергеевич    | Секретарь             | Служба | Служба | Служба | Служба |
| 4     | Смирнов Алексей Алексеевич  | Секретарь             | Служба | Служба | Служба | Служба |
| 5     | Соловьев Дмитрий Дмитриевич | Секретарь             | Служба | Служба | Служба | Служба |
| 6     | Степанов Евгений Евгеньевич | Секретарь             | Служба | Служба | Служба | Служба |
| 7     | Сухов Василий Васильевич    | Секретарь             | Служба | Служба | Служба | Служба |
| 8     | Тихонов Алексей Алексеевич  | Секретарь             | Служба | Служба | Служба | Служба |
| 9     | Толкачев Алексей Алексеевич | Секретарь             | Служба | Служба | Служба | Служба |
| 10    | Трофимов Алексей Алексеевич | Секретарь             | Служба | Служба | Служба | Служба |

1/2024 2/2024 3/2024 4/2024 5/2024 6/2024 7/2024

| AgriConnect: Crop Yield Analysis |           |                   |               |              |              |                     |               |
|----------------------------------|-----------|-------------------|---------------|--------------|--------------|---------------------|---------------|
| Crop Details                     |           | Geographical Data |               | Weather Data |              | Performance Metrics |               |
| Crop ID                          | Crop Name | Region            | Sub-Region    | Temp (°C)    | Humidity (%) | Yield (kg/ha)       | Quality Index |
| 001                              | Wheat     | North             | North-West    | 15.2         | 65           | 3500                | 85            |
| 002                              | Wheat     | North             | North-East    | 16.5         | 68           | 3650                | 87            |
| 003                              | Wheat     | North             | South-West    | 14.8         | 62           | 3400                | 83            |
| 004                              | Wheat     | North             | South-East    | 15.5         | 66           | 3550                | 86            |
| 005                              | Wheat     | South             | South-West    | 17.0         | 70           | 3800                | 89            |
| 006                              | Wheat     | South             | South-East    | 17.5         | 72           | 3900                | 91            |
| 007                              | Wheat     | South             | North-West    | 16.8         | 69           | 3750                | 88            |
| 008                              | Wheat     | South             | North-East    | 17.2         | 71           | 3850                | 90            |
| 009                              | Wheat     | West              | West-North    | 15.0         | 64           | 3450                | 84            |
| 010                              | Wheat     | West              | West-South    | 15.3         | 67           | 3500                | 86            |
| 011                              | Wheat     | West              | East-North    | 15.1         | 63           | 3480                | 84            |
| 012                              | Wheat     | West              | East-South    | 15.4         | 66           | 3520                | 87            |
| 013                              | Wheat     | East              | East-North    | 16.0         | 69           | 3600                | 88            |
| 014                              | Wheat     | East              | East-South    | 16.3         | 71           | 3650                | 90            |
| 015                              | Wheat     | East              | West-North    | 15.8         | 67           | 3580                | 87            |
| 016                              | Wheat     | East              | West-South    | 16.1         | 70           | 3620                | 89            |
| 017                              | Wheat     | Central           | Central-North | 15.6         | 65           | 3530                | 86            |
| 018                              | Wheat     | Central           | Central-South | 15.9         | 68           | 3570                | 88            |
| 019                              | Wheat     | Central           | North-West    | 15.4         | 64           | 3510                | 85            |
| 020                              | Wheat     | Central           | North-East    | 15.7         | 67           | 3540                | 87            |
| 021                              | Wheat     | Central           | South-West    | 15.2         | 63           | 3490                | 84            |
| 022                              | Wheat     | Central           | South-East    | 15.5         | 66           | 3520                | 86            |
| 023                              | Wheat     | Central           | West-North    | 15.1         | 62           | 3470                | 83            |
| 024                              | Wheat     | Central           | West-South    | 15.4         | 65           | 3500                | 85            |
| 025                              | Wheat     | Central           | East-North    | 15.8         | 69           | 3560                | 88            |
| 026                              | Wheat     | Central           | East-South    | 16.1         | 72           | 3590                | 90            |
| 027                              | Wheat     | Central           | North-West    | 15.3         | 64           | 3510                | 85            |
| 028                              | Wheat     | Central           | North-East    | 15.6         | 67           | 3540                | 87            |
| 029                              | Wheat     | Central           | South-West    | 15.0         | 62           | 3460                | 83            |
| 030                              | Wheat     | Central           | South-East    | 15.3         | 65           | 3490                | 85            |
| 031                              | Wheat     | Central           | West-North    | 14.9         | 61           | 3440                | 82            |
| 032                              | Wheat     | Central           | West-South    | 15.2         | 64           | 3470                | 84            |
| 033                              | Wheat     | Central           | East-North    | 15.7         | 68           | 3530                | 87            |
| 034                              | Wheat     | Central           | East-South    | 16.0         | 71           | 3560                | 89            |
| 035                              | Wheat     | Central           | North-West    | 15.5         | 66           | 3520                | 86            |
| 036                              | Wheat     | Central           | North-East    | 15.8         | 69           | 3550                | 88            |
| 037                              | Wheat     | Central           | South-West    | 15.1         | 63           | 3480                | 84            |
| 038                              | Wheat     | Central           | South-East    | 15.4         | 66           | 3510                | 86            |
| 039                              | Wheat     | Central           | West-North    | 14.8         | 61           | 3430                | 82            |
| 040                              | Wheat     | Central           | West-South    | 15.1         | 64           | 3460                | 84            |
| 041                              | Wheat     | Central           | East-North    | 15.6         | 68           | 3520                | 87            |
| 042                              | Wheat     | Central           | East-South    | 15.9         | 71           | 3550                | 89            |
| 043                              | Wheat     | Central           | North-West    | 15.3         | 65           | 3500                | 86            |
| 044                              | Wheat     | Central           | North-East    | 15.6         | 68           | 3530                | 88            |
| 045                              | Wheat     | Central           | South-West    | 15.0         | 62           | 3450                | 83            |
| 046                              | Wheat     | Central           | South-East    | 15.3         | 65           | 3480                | 85            |
| 047                              | Wheat     | Central           | West-North    | 14.7         | 60           | 3410                | 81            |
| 048                              | Wheat     | Central           | West-South    | 15.0         | 63           | 3440                | 83            |
| 049                              | Wheat     | Central           | East-North    | 15.5         | 67           | 3500                | 86            |
| 050                              | Wheat     | Central           | East-South    | 15.8         | 70           | 3530                | 88            |

**Создать пользователя**

Имя

Пароль (составить пароль, чтобы не забыть)

ФИО

Роль

Приветствие

Идентификатор

Вариант

[Отмена](#) [Создать](#)

| No. / No. | Kategori       | Observasi Harian |             |             |             |             |             |             |             | Evaluasi    |             |             |             |             |           |  |  |
|-----------|----------------|------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-----------|--|--|
|           |                | 08.00-09.00      | 09.00-10.00 | 10.00-11.00 | 11.00-12.00 | 12.00-13.00 | 13.00-14.00 | 14.00-15.00 | 15.00-16.00 | 16.00-17.00 | 17.00-18.00 | 18.00-19.00 | 19.00-20.00 | 20.00-21.00 |           |  |  |
| 1         | Indikator 1.1  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 2         | Indikator 1.2  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 3         | Indikator 1.3  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 4         | Indikator 1.4  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 5         | Indikator 1.5  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 6         | Indikator 1.6  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 7         | Indikator 1.7  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 8         | Indikator 1.8  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 9         | Indikator 1.9  |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |
| 10        | Indikator 1.10 |                  |             | 0.00-0.25   | 0.25-0.50   | 0.50-0.75   | 0.75-1.00   | 1.00-1.25   | 1.25-1.50   | 1.50-1.75   | 1.75-2.00   | 2.00-2.25   | 2.25-2.50   | 2.50-2.75   | 2.75-3.00 |  |  |

## Экспорт и отчёты - студенческая ведомость

| гггг | месяц       | 2008/01                                                                                                            | гггг | месяц       | 2008/02                                                               | гггг | месяц       | 2008/03                                                                 | гггг | месяц       | 2008/04                                                               |
|------|-------------|--------------------------------------------------------------------------------------------------------------------|------|-------------|-----------------------------------------------------------------------|------|-------------|-------------------------------------------------------------------------|------|-------------|-----------------------------------------------------------------------|
|      | 8-08-9-05   | Иван<br>Александрович<br>Савин<br>персональный<br>ф.б. №127                                                        |      | 8-08-9-05   |                                                                       |      | 8-08-9-05   | Должностное лицо<br>персонального<br>ф.б. №127<br>Авдеев Ф.Ф. ф.б. №127 |      | 8-08-9-05   |                                                                       |
|      | 9-08-11-05  |                                                                                                                    |      | 9-08-11-05  |                                                                       |      | 9-08-11-05  |                                                                         |      | 9-08-11-05  |                                                                       |
|      | 11-08-13-05 |                                                                                                                    |      | 11-08-13-05 | Должностное лицо<br>персонального<br>ф.б. №127<br>Зубов—Ев. ф.б. №127 |      | 11-08-13-05 |                                                                         |      | 11-08-13-05 |                                                                       |
|      | 13-08-15-05 |                                                                                                                    |      | 13-08-15-05 |                                                                       |      | 13-08-15-05 |                                                                         |      | 13-08-15-05 |                                                                       |
|      | 15-08-17-05 |                                                                                                                    |      | 15-08-17-05 |                                                                       |      | 15-08-17-05 |                                                                         |      | 15-08-17-05 |                                                                       |
|      | 17-08-19-08 |                                                                                                                    |      | 17-08-19-08 |                                                                       |      | 17-08-19-08 |                                                                         |      | 17-08-19-08 |                                                                       |
|      | 19-08-20-02 |                                                                                                                    |      | 19-08-20-02 |                                                                       |      | 19-08-20-02 |                                                                         |      | 19-08-20-02 |                                                                       |
| 01   | 8-08-9-05   | Персональное<br>делопроизводство<br>персонального<br>ф.б. №127<br>Александрович<br>Савин<br>ф.б. №127<br>ф.б. №127 | 01   | 8-08-9-05   |                                                                       | 01   | 8-08-9-05   | Должностное лицо<br>персонального<br>ф.б. №127<br>Авдеев Ф.Ф. ф.б. №127 | 01   | 8-08-9-05   |                                                                       |
|      | 9-08-11-05  |                                                                                                                    |      | 9-08-11-05  |                                                                       |      | 9-08-11-05  |                                                                         |      | 9-08-11-05  |                                                                       |
|      | 11-08-13-05 |                                                                                                                    |      | 11-08-13-05 |                                                                       |      | 11-08-13-05 |                                                                         |      | 11-08-13-05 | Должностное лицо<br>персонального<br>ф.б. №127<br>Зубов—Ев. ф.б. №127 |

## Заключение

Преимущества:

- Автоматический импорт поручений из Word.
- Децентрализация – преподаватели сами планируют занятия.
- Строгая валидация исключает конфликты.
- Поддержка совместных лабораторных занятий.
- Автомаатизация процесса.
- Экспорт в Excel.

Перспективы развития:

- Расширение парсера под другие форматы (Excel, CSV).
- Интеграция с электронным деканатом.
- Напоминания и уведомления о смене расписания.
- Мобильное приложение для студентов.
- Интеграция с системой нагрузки



Спасибо за внимание!

С.А.Сидорова, 2019