Оглавление

[1. Разработка технического задания 2](#_Toc212629210)

[2. Проектирование интерфейса приложения 2](#_Toc212629211)

[3. Проектирование и создание базы данных 5](#_Toc212629212)

[3.1. Разработка структуры БД 5](#_Toc212629213)

[3.2. Создание модели базы данных 8](#_Toc212629214)

[4. Разработка программных модулей 9](#_Toc212629215)

[4.1. Создание сервера 9](#_Toc212629216)

[4.2. Создание клиента 15](#_Toc212629217)

[4.3. Реализация модулей вывода на печать 38](#_Toc212629218)

[4.4. Реализация импорта данных из файла в базу данных 49](#_Toc212629219)

[4.5. Загрузка приложения в репозиторий GitHub 49](#_Toc212629220)

[5. Тестирование и отладка 49](#_Toc212629221)

[5.1. Проведение модульного тестирования 49](#_Toc212629222)

[5.2. Отладка программы 49](#_Toc212629223)

[6. Разработка мобильной версии 49](#_Toc212629224)

[7. Разработка технической документации 49](#_Toc212629225)

[7.1. Руководство пользователя 49](#_Toc212629226)

[7.2. Руководство программиста 49](#_Toc212629227)

[7.3. Руководство системного программиста 49](#_Toc212629228)

[7.4. Методика тестирования и испытания программы 49](#_Toc212629229)

[Заключение 49](#_Toc212629230)

[Список литературы 49](#_Toc212629231)

Введение

Моя предметная область представляет собой учебный учет для технического колледжа, которая автоматизирует процессы управления студентами, группами, предметами и просмотра расписания.

# 1. Разработка технического задания

# 2. Проектирование интерфейса приложения

Макет представляет собой систему, созданную для преподавателей и администраторов, которая упрощает организацию учебного процесса. Приложение позволяет управлять студентами, учебными группами и предметами, а также просматривать расписание занятий. Основные возможности включают добавление новых записей, редактирование существующих данных и удаление устаревшей информации о студентах, группах и предметах. Макет сделан на веб-сайте Figma

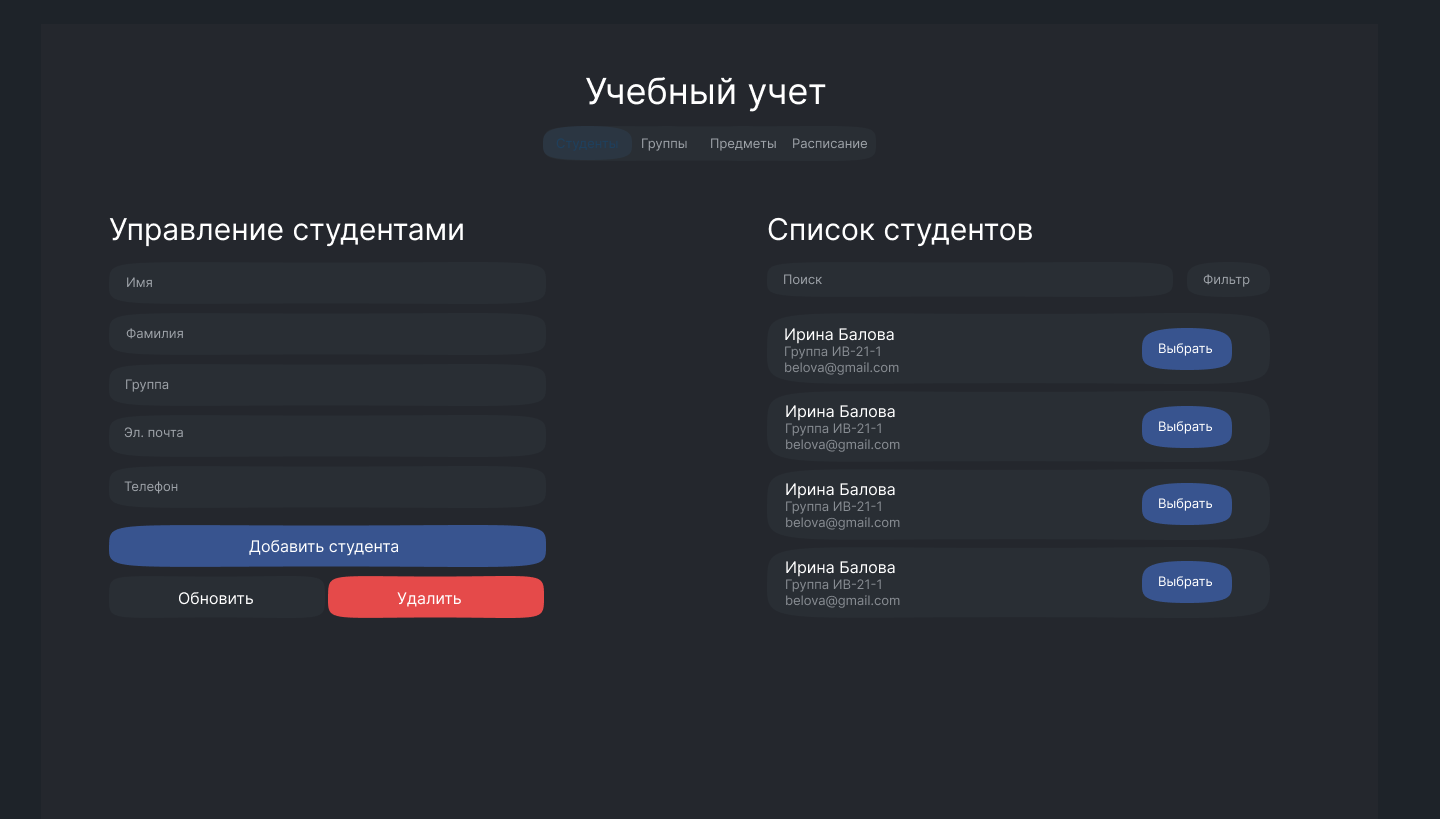


Рис. 1 «Студенты»

На данной странице преподаватель может управлять данными студентов: добавлять новых, редактировать или удалять существующие записи. Также реализована возможность поиска студентов по различным параметрам. В правой части интерфейса отображается список студентов с краткой информацией о каждом из них.

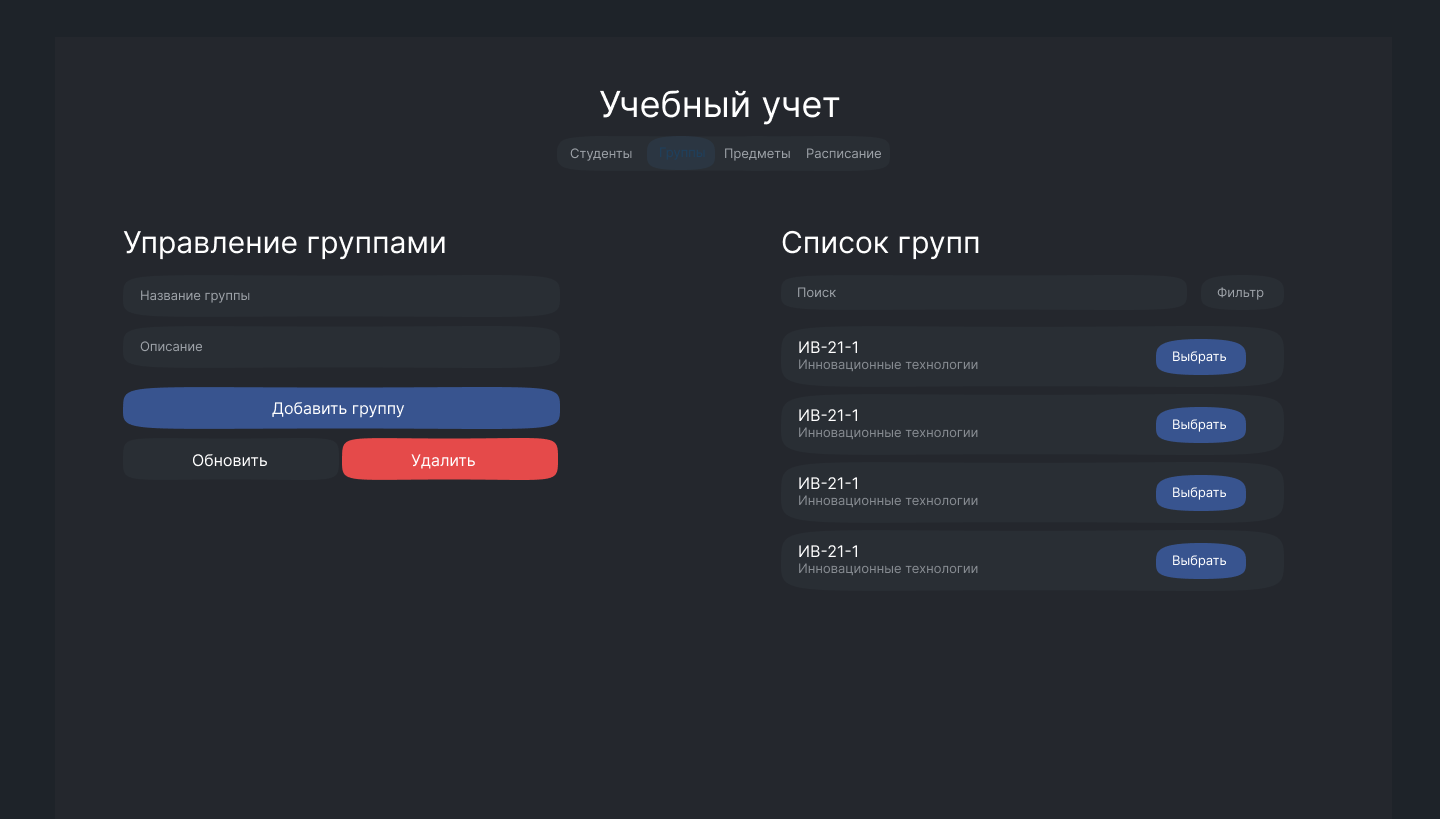


Рис. 2 «Группы»

На этой странице администратор может создавать, изменять и удалять учебные группы. Для каждой группы указываются её название и краткое описание. Справа представлен список всех групп с возможностью поиска, что позволяет быстро находить нужную запись.

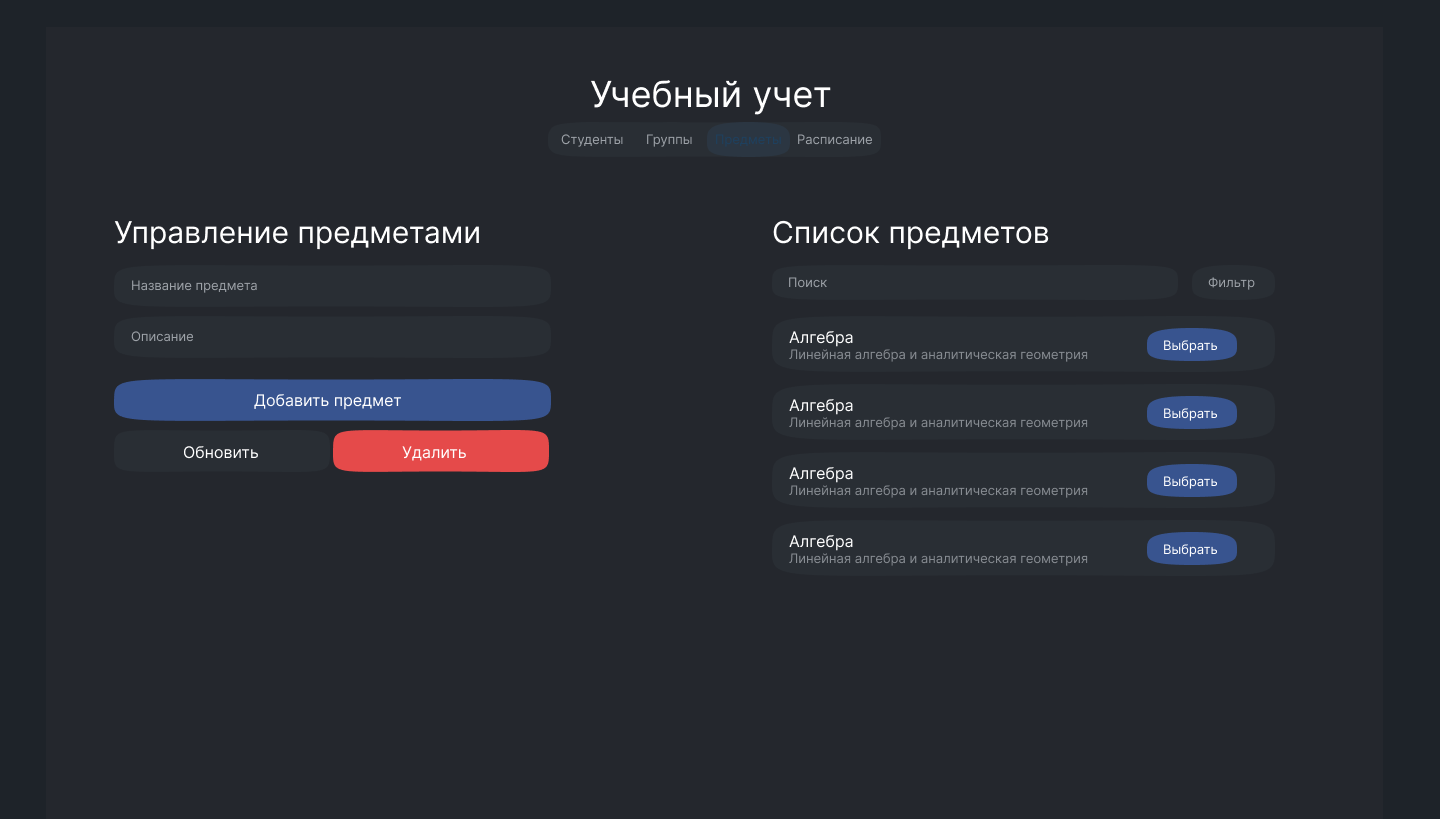


Рис. 3 «Предметы»

На этой странице преподаватель может создавать, изменять и удалять предметы. Для каждого предмета указывается её название и краткое описание. Справа представлен список всех предметов с возможностью поиска, что позволяет быстро находить нужную запись.

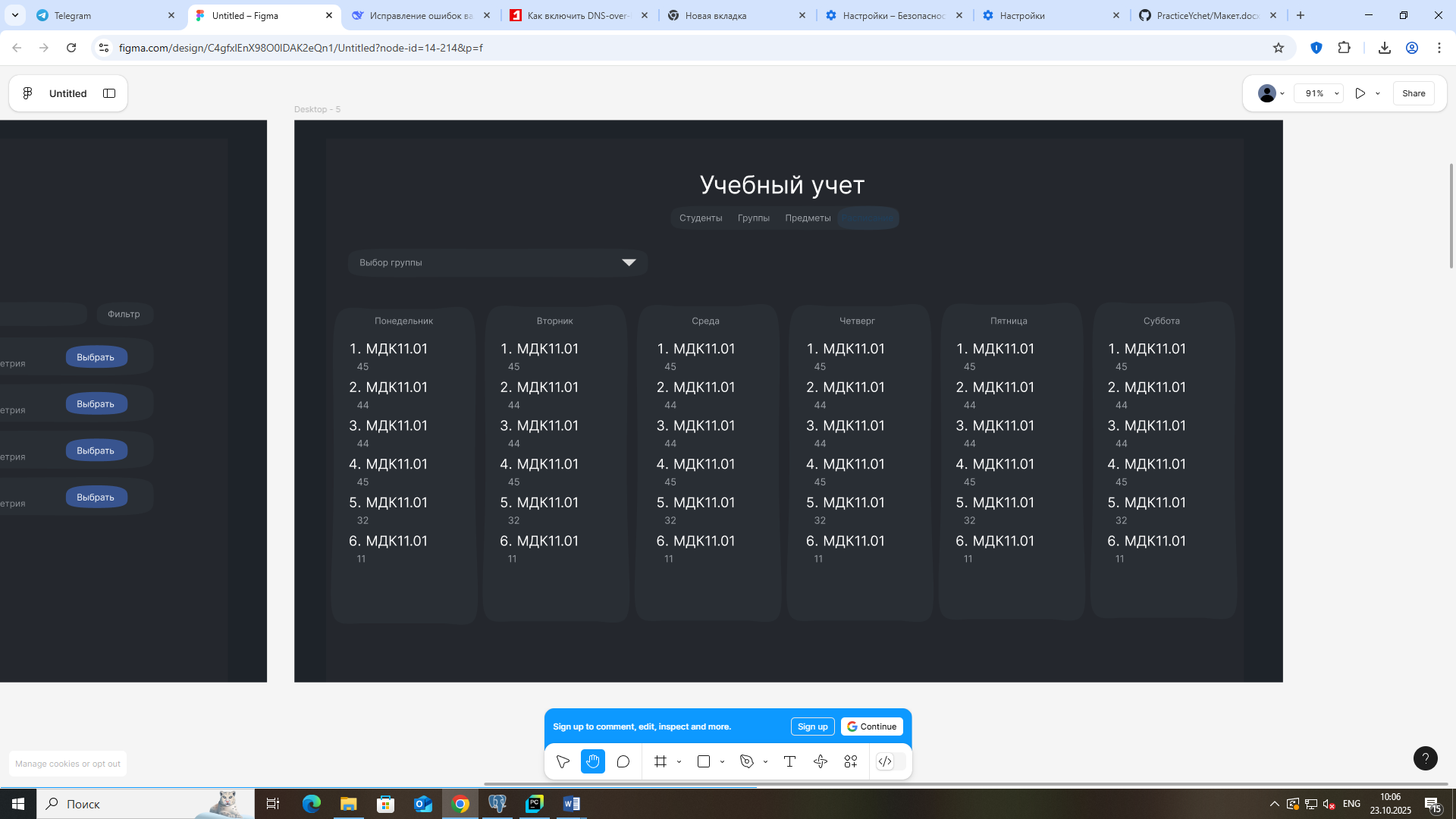


Рис. 4 «Расписание»

Данная страница предназначена для просмотра расписания занятий по выбранной группе. Преподаватель может выбрать нужную группу из выпадающего списка и ознакомиться с расписанием на каждый день недели. Интерфейс структурирован по дням, что обеспечивает удобное восприятие и навигацию.

Достоинства приложения Figma:

* Совместная работа в реальном времени
* Облачное хранилище
* Готовые UI-киты
* Экспорт кода CSS

Недостатки приложения Figma:

* Требует интернет
* Кривая обучения
* Нужно время на изучение интерфейса
* Много функций может запутать новичка
* Ограничения бесплатной версии
* Не всегда точный код CSS

# 3. Проектирование и создание базы данных

## 3.1. Разработка структуры БД

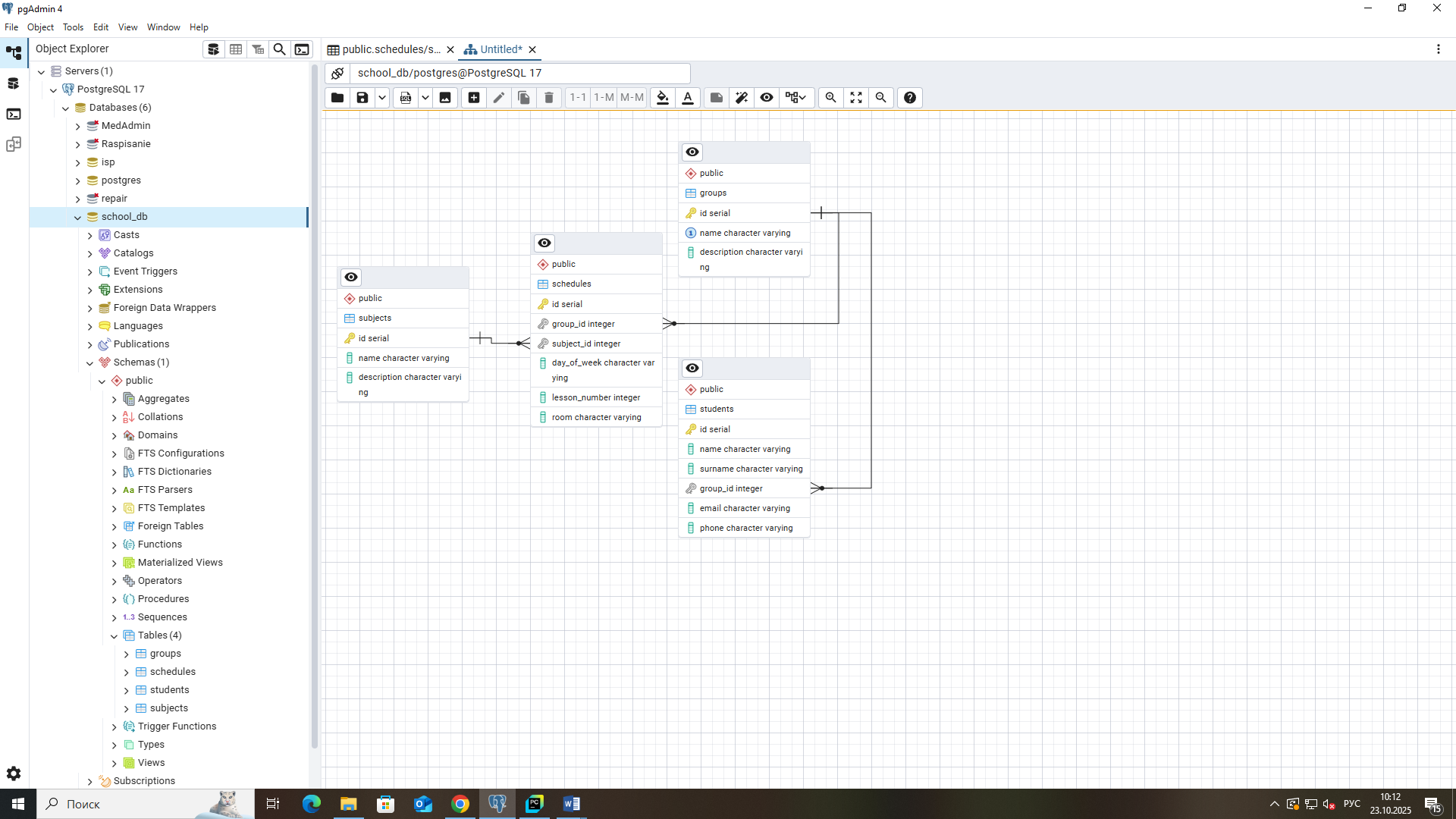


Рис. 5 «ER-диаграмма»

Таблица students предназначена для хранения информации о студентах. В её структуре предусмотрены следующие атрибуты:

* id – первичный ключ. Тип данных integer, not null, primary key, nextval nextval('students\_id\_seq'::regclass)
* name – имя студента. Тип данных character varying, not null
* surname – фамилия студента. Тип данных character varying, not null
* group\_id – внешний ключ таблицы groups. Тип данных integer
* email – почта студента. Тип данных character varying
* phone – телефон студента. Тип данных character varying

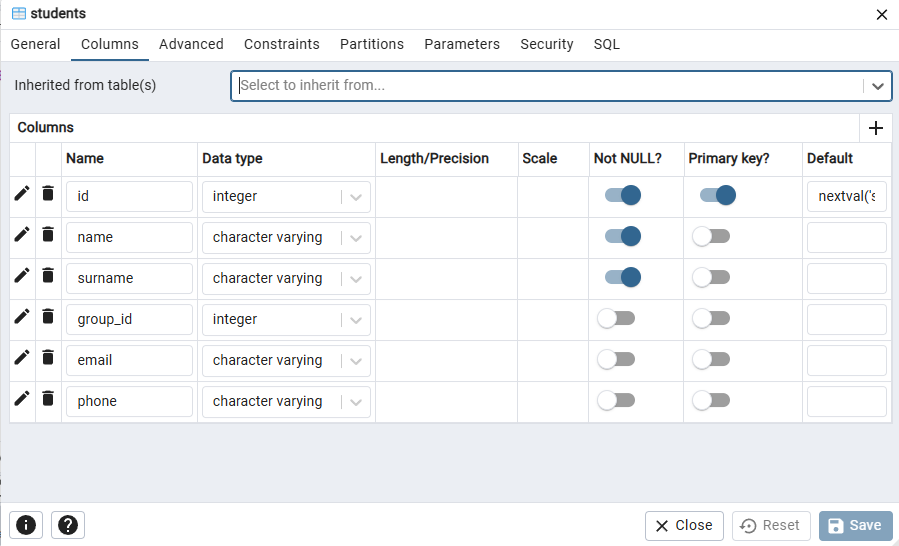


Рис. 6 «Таблица students»

Таблица subjects предназначена для хранения информации о предметах. Она включает следующие атрибуты:

* id – первичный ключ. Тип данных integer, not null, primary key, nextval('subjects\_id\_seq'::regclass)
* name – имя предмета. Тип данных character varying, not null
* description – описание предмета. Тип данных character varying

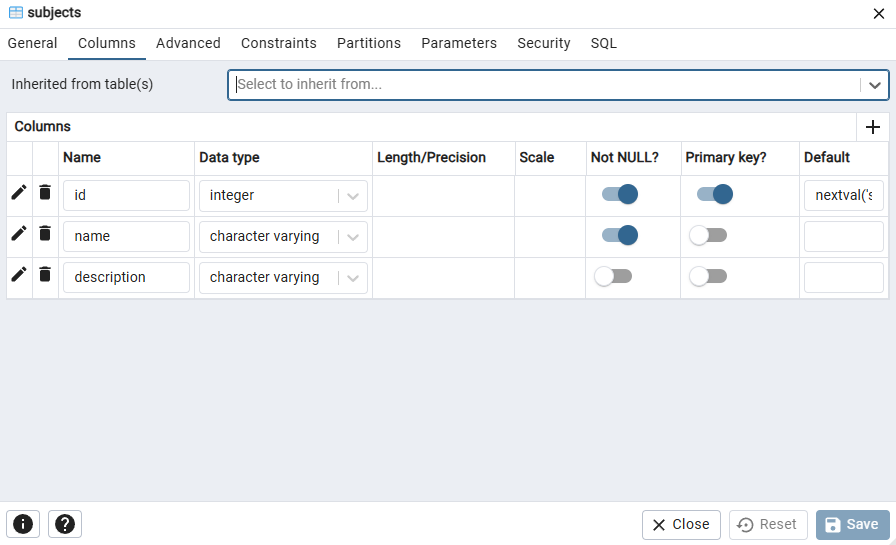


Рис. 7 «Таблица subjects»

Таблица groups служит для хранения сведений об учебных группах. В её структуру входят следующие атрибуты:

* id – первичный ключ. Тип данных integer, not null, primary key, nextval('groups\_id\_seq'::regclass).
* name – имя группы. Тип данных character varying, not null
* description – описание группы. Тип данных character varying

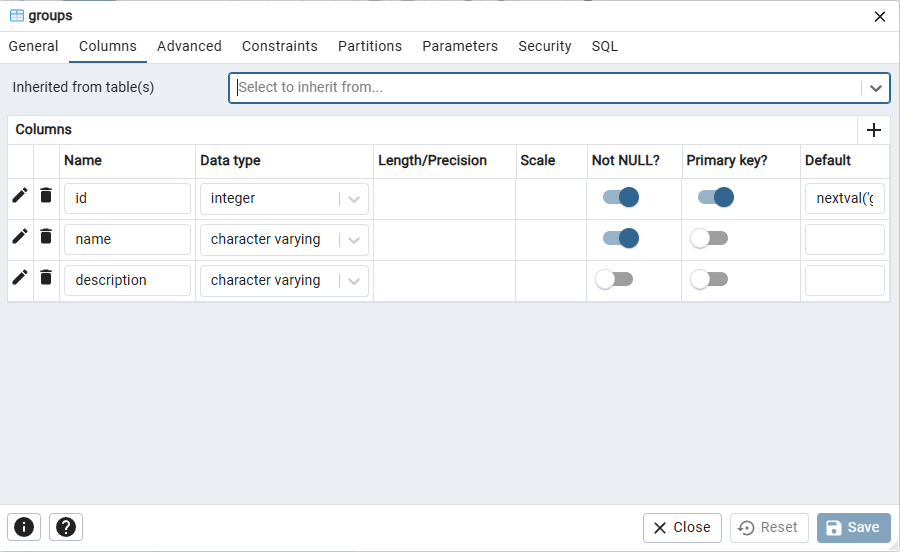


Рис. 8 «Таблица groups»

Таблица schedules используется для формирования расписания занятий. Она содержит следующие атрибуты:

* id – первичный ключ. Тип данных integer, not null, primary key, nextval('schedules\_id\_seq'::regclass)
* group\_id – внешний ключ таблицы groups. Тип данных integer, not null
* subject\_id – внешний ключ таюлицы subjects. Тип данных integer, not null
* day\_of\_week – день недели в таблице расписания. Тип данных character varying, not null
* lesson\_number – номер пары в таблице расписания. Тип данных integer, not null
* room – номер аудитории в таблице расписания. Тип данных character varying

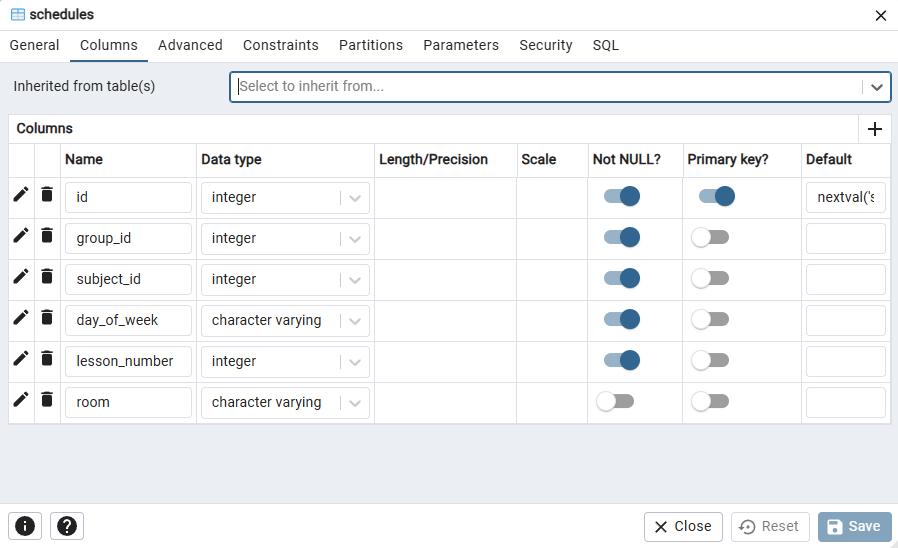


Рис. 9 «Таблица schedules»

## 3.2. Создание модели базы данных

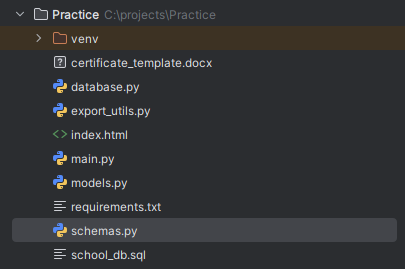


Рис. 10 «Иерархия файлов»

- модуль классов

Листинг 1 – класс Student

class Student(Base):  
 \_\_tablename\_\_ = "students"  
  
 id = Column(Integer, primary\_key=True, index=True)  
 name = Column(String, nullable=False)  
 surname = Column(String, nullable=False)  
 group\_id = Column(Integer, ForeignKey("groups.id"), nullable=True)  
 email = Column(String, nullable=True)  
 phone = Column(String, nullable=True)  
  
 group = relationship("Group", back\_populates="students")

Листинг 2 – класс Group

class Group(Base):  
 \_\_tablename\_\_ = "groups"  
  
 id = Column(Integer, primary\_key=True, index=True)  
 name = Column(String, unique=True, nullable=False)  
 description = Column(String, nullable=True)  
  
 students = relationship("Student", back\_populates="group")  
 schedules = relationship("Schedule", back\_populates="group")

Листинг 3 – класс Subject

class Subject(Base):  
 \_\_tablename\_\_ = "subjects"  
  
 id = Column(Integer, primary\_key=True, index=True)  
 name = Column(String, nullable=False)  
 description = Column(String, nullable=True)  
  
 schedules = relationship("Schedule", back\_populates="subject")

Листинг 4 – класс Schedule

class Schedule(Base):  
 \_\_tablename\_\_ = "schedules"  
  
 id = Column(Integer, primary\_key=True, index=True)  
 group\_id = Column(Integer, ForeignKey("groups.id"), nullable=False)  
 subject\_id = Column(Integer, ForeignKey("subjects.id"), nullable=False)  
 day\_of\_week = Column(String, nullable=False)   
 lesson\_number = Column(Integer, nullable=False)  
 room = Column(String, nullable=True)  
  
 group = relationship("Group", back\_populates="schedules")  
 subject = relationship("Subject", back\_populates="schedules")

# 4. Разработка программных модулей

## 4.1. Создание сервера

- модуль подключения к базе данных

Листинг 5 – подключение к базе данных

DATABASE\_URL = "postgresql://postgres:1234@localhost:5432/school\_db"  
  
*# Создание движка базы данных*engine = create\_engine(  
 DATABASE\_URL,  
 pool\_pre\_ping=True,  
 echo=False  
)  
  
*# Создание сессии*SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
  
*# Базовый класс для моделей*Base = declarative\_base()  
  
*# Зависимость для получения сессии БД*def get\_db():  
 db = SessionLocal()  
 try:  
 yield db  
 finally:  
 db.close()

- модуль функций доступа к данным API (GET, POST, PUT, DELETE)

1. API для студентов

Листинг 6 — Получение списка студентов

@app.get("/api/students", response\_model=List[schemas.Student])  
def get\_students(db: Session = Depends(get\_db)):  
 return db.query(models.Student).all()

Листинг 7 — Создание нового студента

@app.post("/api/students", response\_model=schemas.Student)  
def create\_student(student: schemas.StudentCreate, db: Session = Depends(get\_db)):  
 db\_student = models.Student(\*\*student.dict())  
 db.add(db\_student)  
 db.commit()  
 db.refresh(db\_student)  
 return db\_student

Листинг 8 — Обновление данных студента

@app.put("/api/students/{student\_id}", response\_model=schemas.Student)  
def update\_student(student\_id: int, student: schemas.StudentCreate, db: Session = Depends(get\_db)):  
 db\_student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not db\_student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
 for key, value in student.dict().items():  
 setattr(db\_student, key, value)  
 db.commit()  
 db.refresh(db\_student)  
 return db\_student

Листинг 9 — Удаление студента

@app.delete("/api/students/{student\_id}")  
def delete\_student(student\_id: int, db: Session = Depends(get\_db)):  
 db\_student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not db\_student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
 db.delete(db\_student)  
 db.commit()  
 return {"message": "Студент удален"}

2. API для групп

Листинг 10 — Получение списка групп

@app.get("/api/groups", response\_model=List[schemas.Group])  
def get\_groups(db: Session = Depends(get\_db)):  
 return db.query(models.Group).all()

Листинг 11 — Создание новой группы

@app.post("/api/groups", response\_model=schemas.Group)  
def create\_group(group: schemas.GroupCreate, db: Session = Depends(get\_db)):  
 db\_group = models.Group(\*\*group.dict())  
 db.add(db\_group)  
 db.commit()  
 db.refresh(db\_group)  
 return db\_group

Листинг 12 — Обновление группы

@app.put("/api/groups/{group\_id}", response\_model=schemas.Group)  
def update\_group(group\_id: int, group: schemas.GroupCreate, db: Session = Depends(get\_db)):  
 db\_group = db.query(models.Group).filter(models.Group.id == group\_id).first()  
 if not db\_group:  
 raise HTTPException(status\_code=404, detail="Группа не найдена")  
 for key, value in group.dict().items():  
 setattr(db\_group, key, value)  
 db.commit()  
 db.refresh(db\_group)  
 return db\_group

Листинг 13 — Удаление группы

@app.delete("/api/groups/{group\_id}")  
def delete\_group(group\_id: int, db: Session = Depends(get\_db)):  
 db\_group = db.query(models.Group).filter(models.Group.id == group\_id).first()  
 if not db\_group:  
 raise HTTPException(status\_code=404, detail="Группа не найдена")  
 db.delete(db\_group)  
 db.commit()  
 return {"message": "Группа удалена"}

3. API для предметов

Листинг 14 — Получение списка предметов

@app.get("/api/subjects", response\_model=List[schemas.Subject])  
def get\_subjects(db: Session = Depends(get\_db)):  
 return db.query(models.Subject).all()

Листинг 15 — Создание нового предмета

@app.post("/api/subjects", response\_model=schemas.Subject)  
def create\_subject(subject: schemas.SubjectCreate, db: Session = Depends(get\_db)):  
 db\_subject = models.Subject(\*\*subject.dict())  
 db.add(db\_subject)  
 db.commit()  
 db.refresh(db\_subject)  
 return db\_subject

Листинг 16 — Обновление предмета

@app.put("/api/subjects/{subject\_id}", response\_model=schemas.Subject)  
def update\_subject(subject\_id: int, subject: schemas.SubjectCreate, db: Session = Depends(get\_db)):  
 db\_subject = db.query(models.Subject).filter(models.Subject.id == subject\_id).first()  
 if not db\_subject:  
 raise HTTPException(status\_code=404, detail="Предмет не найден")  
 for key, value in subject.dict().items():  
 setattr(db\_subject, key, value)  
 db.commit()  
 db.refresh(db\_subject)  
 return db\_subject

Листинг 17 — Удаление предмета

@app.delete("/api/subjects/{subject\_id}")  
def delete\_subject(subject\_id: int, db: Session = Depends(get\_db)):  
 db\_subject = db.query(models.Subject).filter(models.Subject.id == subject\_id).first()  
 if not db\_subject:  
 raise HTTPException(status\_code=404, detail="Предмет не найден")  
 db.delete(db\_subject)  
 db.commit()  
 return {"message": "Предмет удален"}

4. API для расписания

Листинг 18 — Получение расписания

@app.get("/api/schedule", response\_model=List[schemas.Schedule])  
def get\_schedule(group\_id: int = None, db: Session = Depends(get\_db)):  
 query = db.query(models.Schedule)  
 if group\_id:  
 query = query.filter(models.Schedule.group\_id == group\_id)  
 return query.all()

Листинг 19 — Создание записи расписания

@app.post("/api/schedule", response\_model=schemas.Schedule)  
def create\_schedule(schedule: schemas.ScheduleCreate, db: Session = Depends(get\_db)):  
 db\_schedule = models.Schedule(\*\*schedule.dict())  
 db.add(db\_schedule)  
 db.commit()  
 db.refresh(db\_schedule)  
 return db\_schedule

Листинг 20 — Обновление записи расписания

@app.put("/api/schedule/{schedule\_id}", response\_model=schemas.Schedule)  
def update\_schedule(schedule\_id: int, schedule: schemas.ScheduleCreate, db: Session = Depends(get\_db)):  
 db\_schedule = db.query(models.Schedule).filter(models.Schedule.id == schedule\_id).first()  
 if not db\_schedule:  
 raise HTTPException(status\_code=404, detail="Расписание не найдено")  
 for key, value in schedule.dict().items():  
 setattr(db\_schedule, key, value)  
 db.commit()  
 db.refresh(db\_schedule)  
 return db\_schedule

Листинг 21 — Удаление записи расписания

@app.delete("/api/schedule/{schedule\_id}")  
def delete\_schedule(schedule\_id: int, db: Session = Depends(get\_db)):  
 db\_schedule = db.query(models.Schedule).filter(models.Schedule.id == schedule\_id).first()  
 if not db\_schedule:  
 raise HTTPException(status\_code=404, detail="Расписание не найдено")  
 db.delete(db\_schedule)  
 db.commit()  
 return {"message": "Расписание удалено"}

5. API для вывода на печать

Листинг 22 — Экспорт справки студента в Word

@app.get("/api/export/student/{student\_id}/certificate-word")  
def export\_student\_certificate\_word(student\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт справки студента в Word"""* student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
  
 group\_name = None  
 if student.group\_id:  
 group = db.query(models.Group).filter(models.Group.id == student.group\_id).first()  
 group\_name = group.name if group else None  
  
 buffer = create\_student\_certificate(student, group\_name)  
  
 *# Безопасное имя файла без русских символов* filename = f"certificate\_{student.surname}\_{student.name}.docx"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/vnd.openxmlformats-officedocument.wordprocessingml.document",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

Листинг 23 — Экспорт расписания в Excel

@app.get("/api/export/schedule/{group\_id}/excel")  
def export\_schedule\_excel(group\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт расписания группы в Excel"""* group = db.query(models.Group).filter(models.Group.id == group\_id).first()  
 if not group:  
 raise HTTPException(status\_code=404, detail="Группа не найдена")  
  
 schedules = db.query(models.Schedule).filter(models.Schedule.group\_id == group\_id).all()  
 subjects = db.query(models.Subject).all()  
 subjects\_dict = {s.id: s.name for s in subjects}  
  
 buffer = create\_schedule\_excel(group, schedules, subjects\_dict)  
  
 *# Безопасное имя файла без русских символов* filename = f"schedule\_{group.name}.xlsx"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

Листинг 24 — Экспорт справки студента в PDF

@app.get("/api/export/student/{student\_id}/certificate-pdf")  
def export\_student\_certificate\_pdf(student\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт справки студента в PDF"""* student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
  
 group\_name = None  
 if student.group\_id:  
 group = db.query(models.Group).filter(models.Group.id == student.group\_id).first()  
 group\_name = group.name if group else None  
  
 buffer = create\_student\_certificate\_pdf(student, group\_name)  
  
 *# Безопасное имя файла без русских символов* filename = f"certificate\_{student.surname}\_{student.name}.pdf"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/pdf",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

## 4.2. Создание клиента

В проекте присутствует единый HTML-файл index.html, который содержит все страницы системы учебного учета технического колледжа. Интерфейс реализован по принципу Single Page Application (SPA) - все страницы находятся в одном файле и переключаются через JavaScript без перезагрузки.

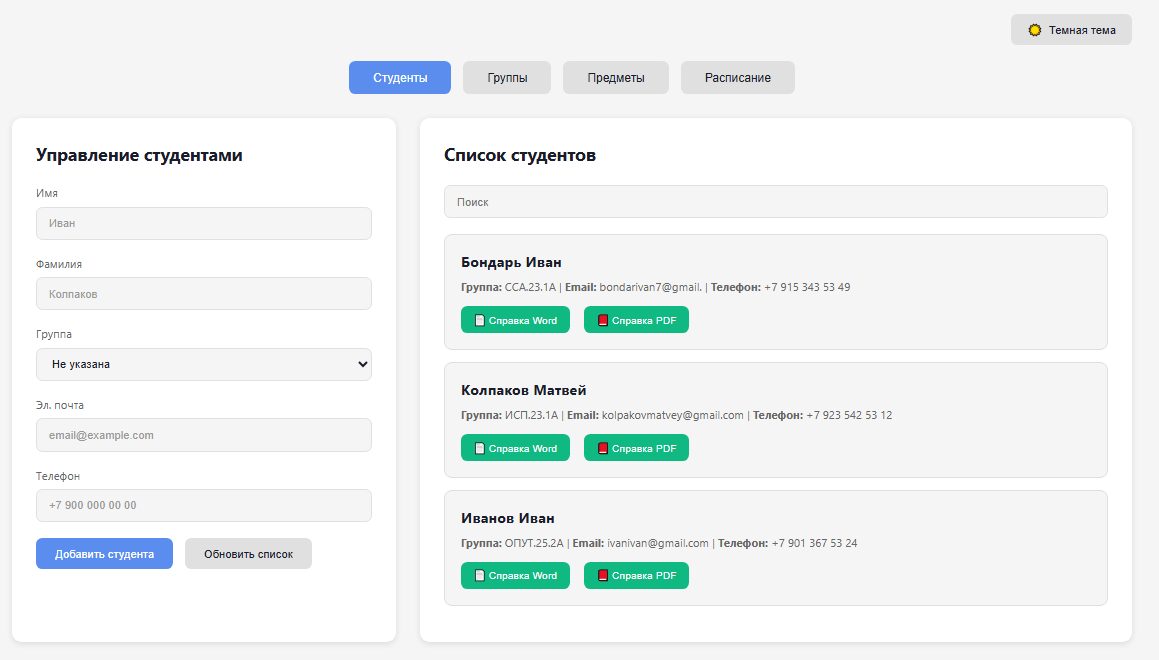


Рис. 11 «Страница Студенты»

Страница предназначена для управления списком студентов колледжа. Содержит форму добавления нового студента и список всех зарегистрированных студентов с возможностью экспорта справок.

Листинг 25 – код страницы студенты

*<!-- СЕКЦИЯ: СТУДЕНТЫ -->*<section id="students" class="section active">  
 <h2 class="section-title">Управление студентами</h2>  
   
 *<!-- Панель управления -->* <div class="controls-panel">  
 <button class="btn btn-primary" onclick="openModal('studentModal')">  
 + Добавить студента  
 </button>  
 </div>  
   
 *<!-- Список студентов в виде карточек -->* <div id="studentsList" class="students-grid">  
 *<!-- Карточка студента (пример) -->* <div class="student-card">  
 <div class="student-header">  
 <h3 class="student-name">Бондарь Иван</h3>  
 </div>  
 <div class="student-info">  
 <p><strong>Группа:</strong> ССА.23.1А</p>  
 <p><strong>Email:</strong> bondarivan7@gmail.i</p>  
 <p><strong>Телефон:</strong> +7 915 343 53 49</p>  
 </div>  
 <div class="student-actions">  
 <button class="btn btn-success">  
 📄 Справка Word  
 </button>  
 <button class="btn btn-danger">  
 📄 Справка PDF  
 </button>  
 </div>  
 </div>  
   
 *<!-- Другие карточки студентов -->* <div class="student-card">  
 <div class="student-header">  
 <h3 class="student-name">Колпаков Матвей</h3>  
 </div>  
 <div class="student-info">  
 <p><strong>Группа:</strong> ИСП.23.1А</p>  
 <p><strong>Email:</strong> kolpakovmatvey@gmail.com</p>  
 <p><strong>Телефон:</strong> +7 923 542 53 12</p>  
 </div>  
 <div class="student-actions">  
 <button class="btn btn-success">  
 📄 Справка Word  
 </button>  
 <button class="btn btn-danger">  
 📄 Справка PDF  
 </button>  
 </div>  
 </div>  
 </div>  
   
 *<!-- Поиск по студентам -->* <div class="search-panel">  
 <input type="text" class="form-control"   
 placeholder="Поиск по имени или группе..."   
 onkeyup="filterStudents()">  
 </div>  
</section>  
  
*<!-- Модальное окно добавления студента -->*<div id="studentModal" class="modal">  
 <div class="modal-content">  
 <h3>Управление студентами</h3>  
 <form id="studentForm">  
 <div class="form-group">  
 <label>Имя</label>  
 <input type="text" name="name"   
 class="form-control"   
 placeholder="Иван" required>  
 </div>  
   
 <div class="form-group">  
 <label>Фамилия</label>  
 <input type="text" name="surname"   
 class="form-control"   
 placeholder="Колпаков" required>  
 </div>  
   
 <div class="form-group">  
 <label>Группа</label>  
 <select name="group\_id" class="form-control">  
 <option value="">Не указана</option>  
 *<!-- Загружается динамически -->* </select>  
 </div>  
   
 <div class="form-group">  
 <label>Эл. почта</label>  
 <input type="email" name="email"   
 class="form-control"   
 placeholder="email@example.com">  
 </div>  
   
 <div class="form-group">  
 <label>Телефон</label>  
 <input type="tel" name="phone"   
 class="form-control"   
 placeholder="+7 900 000 00 00">  
 </div>  
   
 <div class="modal-actions">  
 <button type="submit" class="btn btn-primary">  
 Добавить студента  
 </button>  
 <button type="button" class="btn btn-secondary"   
 onclick="closeModal('studentModal')">  
 Обновить список  
 </button>  
 </div>  
 </form>  
 </div>  
</div>

Листинг 26 – функции страницы студенты

// === СТУДЕНТЫ ===  
async function loadStudents() {  
 const response = await fetch(`${API\_URL}/students`);  
 const students = await response.json();  
 const groupsResponse = await fetch(`${API\_URL}/groups`);  
 groups = await groupsResponse.json();  
  
 const studentsList = document.getElementById('studentsList');  
 studentsList.innerHTML = students.map(student => {  
 const group = groups.find(g => g.id === student.group\_id);  
 return `  
 <div class="student-card" onclick="selectStudent(${student.id}, event)">  
 <h3>${student.surname} ${student.name}</h3>  
 <p><strong>Группа:</strong> ${group ? group.name : 'Не указана'} | <strong>Email:</strong> ${student.email || 'Не указан'} | <strong>Телефон:</strong> ${student.phone || 'Не указан'}</p>  
 <div class="actions">  
 <button class="btn btn-success" onclick="exportWordCertificate(${student.id}, event)">📄 Справка Word</button>  
 <button class="btn btn-info" onclick="exportPdfCertificate(${student.id}, event)">📕 Справка PDF</button>  
 </div>  
 </div>  
 `;  
 }).join('');  
  
 // Обновление select группы  
 const groupSelect = document.getElementById('studentGroup');  
 groupSelect.innerHTML = '<option value="">Не указана</option>' +  
 groups.map(g => `<option value="${g.id}">${g.name}</option>`).join('');  
}  
  
function selectStudent(id, event) {  
 // Если клик был по кнопке, не выбираем студента  
 if (event.target.tagName === 'BUTTON') {  
 return;  
 }  
  
 // Снимаем выделение со всех карточек  
 document.querySelectorAll('.student-card').forEach(card => {  
 card.classList.remove('selected');  
 });  
  
 // Выделяем текущую карточку  
 event.currentTarget.classList.add('selected');  
  
 // Загружаем данные студента в форму  
 fetch(`${API\_URL}/students`)  
 .then(res => res.json())  
 .then(students => {  
 const student = students.find(s => s.id === id);  
 if (student) {  
 document.getElementById('studentName').value = student.name;  
 document.getElementById('studentSurname').value = student.surname;  
 document.getElementById('studentGroup').value = student.group\_id || '';  
 document.getElementById('studentEmail').value = student.email || '';  
 document.getElementById('studentPhone').value = student.phone || '';  
 currentStudentId = id;  
 }  
 });  
}  
  
async function addStudent() {  
 const data = {  
 name: document.getElementById('studentName').value,  
 surname: document.getElementById('studentSurname').value,  
 group\_id: parseInt(document.getElementById('studentGroup').value) || null,  
 email: document.getElementById('studentEmail').value || null,  
 phone: document.getElementById('studentPhone').value || null  
 };  
  
 if (!data.name || !data.surname) {  
 alert('Заполните имя и фамилию!');  
 return;  
 }  
  
 if (currentStudentId) {  
 await fetch(`${API\_URL}/students/${currentStudentId}`, {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 currentStudentId = null;  
 } else {  
 await fetch(`${API\_URL}/students`, {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 }  
  
 clearStudentForm();  
 loadStudents();  
}  
  
function clearStudentForm() {  
 document.getElementById('studentName').value = '';  
 document.getElementById('studentSurname').value = '';  
 document.getElementById('studentGroup').value = '';  
 document.getElementById('studentEmail').value = '';  
 document.getElementById('studentPhone').value = '';  
 currentStudentId = null;  
  
 // Снимаем выделение  
 document.querySelectorAll('.student-card').forEach(card => {  
 card.classList.remove('selected');  
 });  
  
 loadStudents();  
}  
  
function filterStudents() {  
 const search = document.getElementById('searchStudent').value.toLowerCase();  
 const cards = document.querySelectorAll('.student-card');  
 cards.forEach(card => {  
 const text = card.textContent.toLowerCase();  
 card.style.display = text.includes(search) ? 'block' : 'none';  
 });  
}  
  
function exportWordCertificate(studentId, event) {  
 event.stopPropagation();  
 window.open(`${API\_URL}/export/student/${studentId}/certificate-word`, '\_blank');  
}  
  
function exportPdfCertificate(studentId, event) {  
 event.stopPropagation();  
 window.open(`${API\_URL}/export/student/${studentId}/certificate-pdf`, '\_blank');  
}

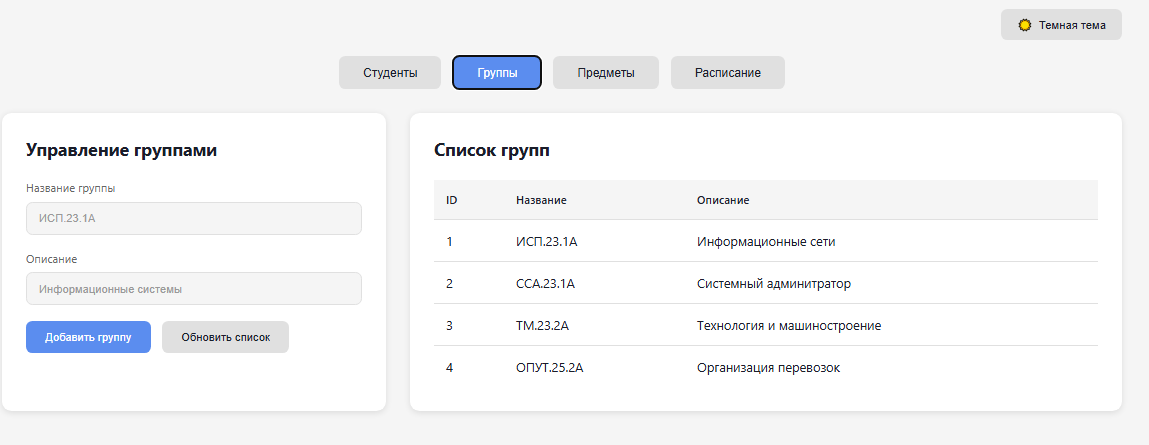


Рис. 12 «Страница Группы»

Страница управления учебными группами. Слева расположена форма для создания новой группы, справа - таблица со списком всех групп.

Листинг 27 – код страницы группы

*<!-- СЕКЦИЯ: ГРУППЫ -->*<section id="groups" class="section">  
 <div class="two-column-layout">  
 *<!-- Левая колонка: Форма управления -->* <div class="left-panel">  
 <h2 class="section-title">Управление группами</h2>  
   
 <form id="groupForm">  
 <div class="form-group">  
 <label>Название группы</label>  
 <input type="text" name="name"   
 class="form-control"   
 placeholder="ИСП.23.1А" required>  
 </div>  
   
 <div class="form-group">  
 <label>Описание</label>  
 <textarea name="description"   
 class="form-control"   
 rows="4"  
 placeholder="Информационные системы"></textarea>  
 </div>  
   
 <div class="form-actions">  
 <button type="submit" class="btn btn-primary">  
 Добавить группу  
 </button>  
 <button type="button" class="btn btn-secondary"  
 onclick="resetGroupForm()">  
 Обновить список  
 </button>  
 </div>  
 </form>  
 </div>  
   
 *<!-- Правая колонка: Список групп -->* <div class="right-panel">  
 <h2 class="section-title">Список групп</h2>  
   
 <table class="data-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Название</th>  
 <th>Описание</th>  
 </tr>  
 </thead>  
 <tbody id="groupsTableBody">  
 <tr>  
 <td>1</td>  
 <td>ИСП.23.1А</td>  
 <td>Информационные сети</td>  
 </tr>  
 <tr>  
 <td>2</td>  
 <td>ССА.23.1А</td>  
 <td>Системный администратор</td>  
 </tr>  
 <tr>  
 <td>3</td>  
 <td>ТМ.23.2А</td>  
 <td>Технология и машиностроение</td>  
 </tr>  
 <tr>  
 <td>4</td>  
 <td>ОПУ.25.2А</td>  
 <td>Организация перевозок</td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
</section>

Листинг 28 – функции страницы группы

// === ГРУППЫ ===  
async function loadGroups() {  
 const response = await fetch(`${API\_URL}/groups`);  
 groups = await response.json();  
  
 const groupsList = document.getElementById('groupsList');  
 groupsList.innerHTML = groups.map(group => `  
 <tr onclick="selectGroup(${group.id})" class="${currentGroupId === group.id ? 'selected' : ''}">  
 <td>${group.id}</td>  
 <td>${group.name}</td>  
 <td>${group.description || '-'}</td>  
 </tr>  
 `).join('');  
}  
  
function selectGroup(id) {  
 // Снимаем выделение со всех строк  
 document.querySelectorAll('#groupsList tr').forEach(row => {  
 row.classList.remove('selected');  
 });  
  
 // Выделяем текущую строку  
 event.currentTarget.classList.add('selected');  
  
 // Загружаем данные группы в форму  
 const group = groups.find(g => g.id === id);  
 if (group) {  
 document.getElementById('groupName').value = group.name;  
 document.getElementById('groupDescription').value = group.description || '';  
 currentGroupId = id;  
 }  
}  
  
async function addGroup() {  
 const data = {  
 name: document.getElementById('groupName').value,  
 description: document.getElementById('groupDescription').value || null  
 };  
  
 if (!data.name) {  
 alert('Заполните название группы!');  
 return;  
 }  
  
 if (currentGroupId) {  
 await fetch(`${API\_URL}/groups/${currentGroupId}`, {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 currentGroupId = null;  
 } else {  
 await fetch(`${API\_URL}/groups`, {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 }  
  
 clearGroupForm();  
 loadGroups();  
}  
  
function clearGroupForm() {  
 document.getElementById('groupName').value = '';  
 document.getElementById('groupDescription').value = '';  
 currentGroupId = null;  
 loadGroups();  
}

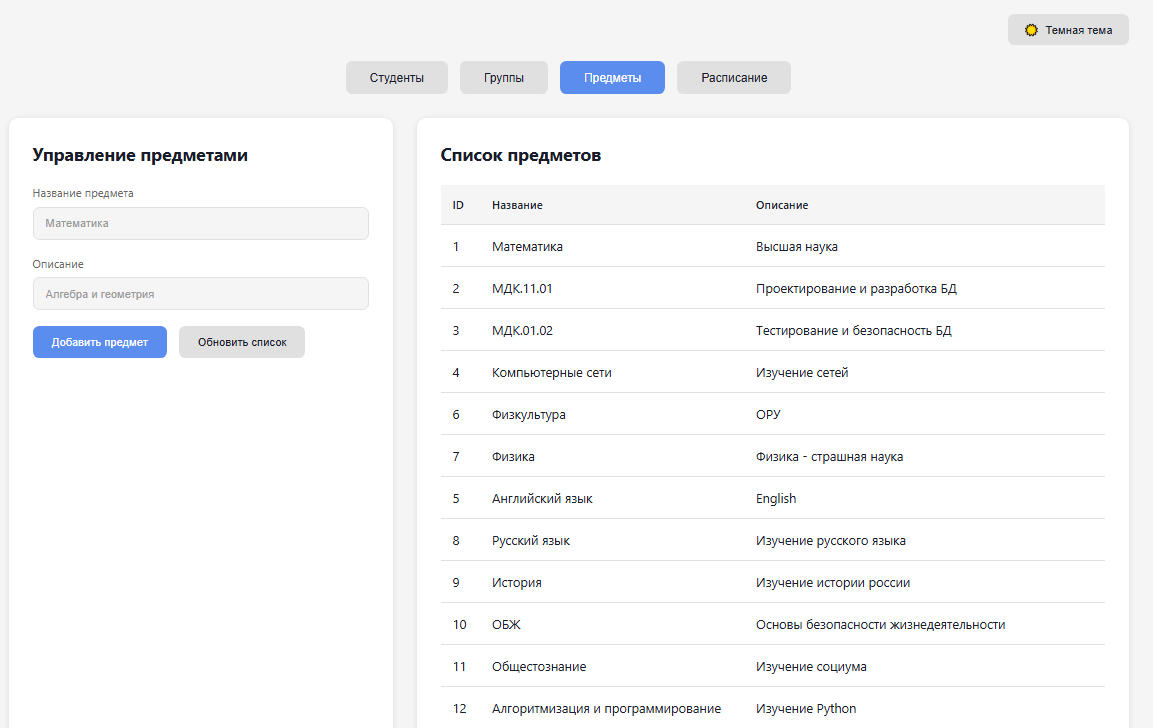


Рис. 13 «Страница Предметы»

Страница для управления учебными дисциплинами. Аналогична странице групп: слева форма добавления, справа таблица с предметами.

Листинг 29 – код страницы предметов

*<!-- СЕКЦИЯ: ПРЕДМЕТЫ -->*<section id="subjects" class="section">  
 <div class="two-column-layout">  
 *<!-- Левая колонка: Форма управления -->* <div class="left-panel">  
 <h2 class="section-title">Управление предметами</h2>  
   
 <form id="subjectForm">  
 <div class="form-group">  
 <label>Название предмета</label>  
 <input type="text" name="name"   
 class="form-control"   
 placeholder="Математика" required>  
 </div>  
   
 <div class="form-group">  
 <label>Описание</label>  
 <textarea name="description"   
 class="form-control"   
 rows="4"  
 placeholder="Алгебра и геометрия"></textarea>  
 </div>  
   
 <div class="form-actions">  
 <button type="submit" class="btn btn-primary">  
 Добавить предмет  
 </button>  
 <button type="button" class="btn btn-secondary"  
 onclick="resetSubjectForm()">  
 Обновить список  
 </button>  
 </div>  
 </form>  
 </div>  
   
 *<!-- Правая колонка: Список предметов -->* <div class="right-panel">  
 <h2 class="section-title">Список предметов</h2>  
   
 <table class="data-table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Название</th>  
 <th>Описание</th>  
 </tr>  
 </thead>  
 <tbody id="subjectsTableBody">  
 <tr>  
 <td>1</td>  
 <td>Математика</td>  
 <td>Высшая наука</td>  
 </tr>  
 <tr>  
 <td>2</td>  
 <td>МДК.11.01</td>  
 <td>Проектирование и разработка БД</td>  
 </tr>  
 <tr>  
 <td>3</td>  
 <td>МДК.01.02</td>  
 <td>Тестирование и безопасность БД</td>  
 </tr>  
 <tr>  
 <td>4</td>  
 <td>Компьютерные сети</td>  
 <td>Изучение сетей</td>  
 </tr>  
 <tr>  
 <td>6</td>  
 <td>Физкультура</td>  
 <td>ОРУ</td>  
 </tr>  
 <tr>  
 <td>7</td>  
 <td>Физика</td>  
 <td>Физика</td>  
 </tr>  
 <tr>  
 <td>5</td>  
 <td>Английский язык</td>  
 <td>English</td>  
 </tr>  
 <tr>  
 <td>8</td>  
 <td>Русский язык</td>  
 <td>Изучение русского языка</td>  
 </tr>  
 <tr>  
 <td>9</td>  
 <td>История</td>  
 <td>Изучение истории россии</td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
</section>

Листинг 30 – функции страницы предметов

// === ПРЕДМЕТЫ ===  
async function loadSubjects() {  
 const response = await fetch(`${API\_URL}/subjects`);  
 subjects = await response.json();  
  
 const subjectsList = document.getElementById('subjectsList');  
 subjectsList.innerHTML = subjects.map(subject => `  
 <tr onclick="selectSubject(${subject.id})" class="${currentSubjectId === subject.id ? 'selected' : ''}">  
 <td>${subject.id}</td>  
 <td>${subject.name}</td>  
 <td>${subject.description || '-'}</td>  
 </tr>  
 `).join('');  
}  
  
function selectSubject(id) {  
 // Снимаем выделение со всех строк  
 document.querySelectorAll('#subjectsList tr').forEach(row => {  
 row.classList.remove('selected');  
 });  
  
 // Выделяем текущую строку  
 event.currentTarget.classList.add('selected');  
  
 // Загружаем данные предмета в форму  
 const subject = subjects.find(s => s.id === id);  
 if (subject) {  
 document.getElementById('subjectName').value = subject.name;  
 document.getElementById('subjectDescription').value = subject.description || '';  
 currentSubjectId = id;  
 }  
}  
  
async function addSubject() {  
 const data = {  
 name: document.getElementById('subjectName').value,  
 description: document.getElementById('subjectDescription').value || null  
 };  
  
 if (!data.name) {  
 alert('Заполните название предмета!');  
 return;  
 }  
  
 if (currentSubjectId) {  
 await fetch(`${API\_URL}/subjects/${currentSubjectId}`, {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 currentSubjectId = null;  
 } else {  
 await fetch(`${API\_URL}/subjects`, {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(data)  
 });  
 }  
  
 clearSubjectForm();  
 loadSubjects();  
}  
  
function clearSubjectForm() {  
 document.getElementById('subjectName').value = '';  
 document.getElementById('subjectDescription').value = '';  
 currentSubjectId = null;  
 loadSubjects();  
}

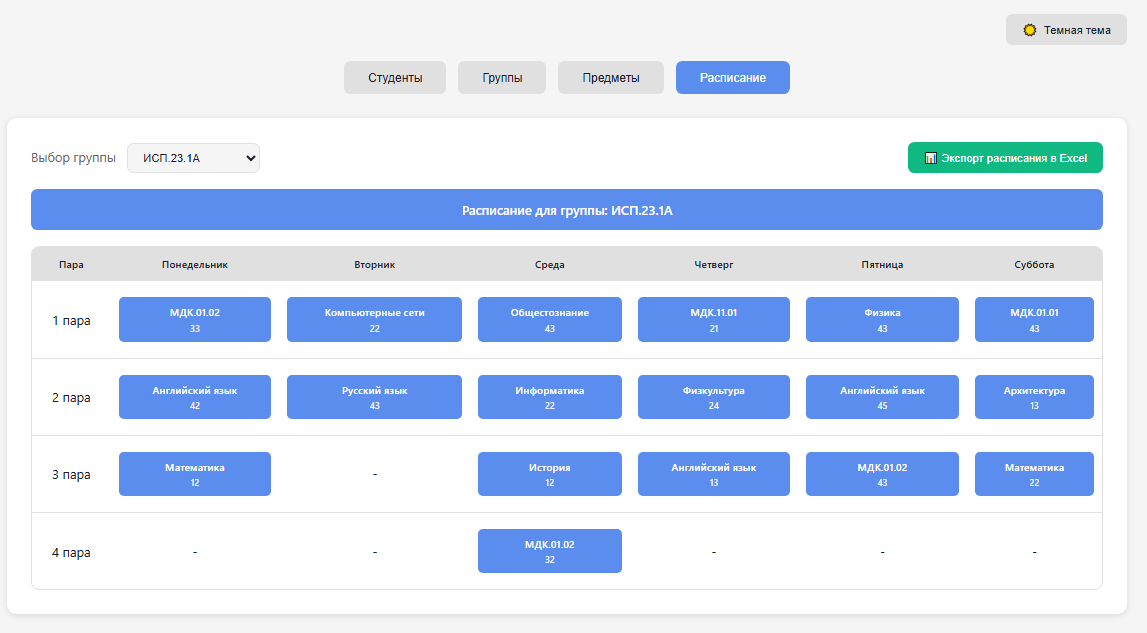


Рис. 14 «Страница Расписание»

Главная страница системы - визуальное отображение расписания занятий. Содержит выпадающий список для выбора группы, кнопку экспорта в Excel и сетку расписания по дням недели и парам.

Листинг 31 – код страницы расписания

*<!-- СЕКЦИЯ: РАСПИСАНИЕ -->*<section id="schedule" class="section">  
 <div class="schedule-container">  
 *<!-- Панель управления расписанием -->* <div class="schedule-controls">  
 <div class="form-group">  
 <label>Выбор группы:</label>  
 <select id="scheduleGroupSelect"   
 class="form-control dark-select"  
 onchange="loadSchedule()">  
 <option value="1">ИСП.23.1А</option>  
 <option value="2">ССА.23.1А</option>  
 <option value="3">ТМ.23.2А</option>  
 <option value="4">ОПУ.25.2А</option>  
 </select>  
 </div>  
   
 <button class="btn btn-success" onclick="exportScheduleExcel()">  
 📊 Экспорт расписания в Excel  
 </button>  
 </div>  
   
 *<!-- Заголовок расписания -->* <div class="schedule-header">  
 <h2 class="schedule-title">  
 Расписание для группы: ИСП.23.1А  
 </h2>  
 </div>  
   
 *<!-- Сетка расписания -->* <div class="schedule-grid">  
 *<!-- Шапка таблицы -->* <div class="schedule-cell header-cell">Пара</div>  
 <div class="schedule-cell header-cell">Понедельник</div>  
 <div class="schedule-cell header-cell">Вторник</div>  
 <div class="schedule-cell header-cell">Среда</div>  
 <div class="schedule-cell header-cell">Четверг</div>  
 <div class="schedule-cell header-cell">Пятница</div>  
 <div class="schedule-cell header-cell">Суббота</div>  
   
 *<!-- 1 ПАРА -->* <div class="schedule-cell lesson-number">1 пара</div>  
   
 *<!-- Понедельник, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">МДК.01.02</div>  
 <div class="lesson-room">33</div>  
 </div>  
   
 *<!-- Вторник, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Компьютерные сети</div>  
 <div class="lesson-room">22</div>  
 </div>  
   
 *<!-- Среда, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Обществознание</div>  
 <div class="lesson-room">43</div>  
 </div>  
   
 *<!-- Четверг, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">МДК.11.01</div>  
 <div class="lesson-room">21</div>  
 </div>  
   
 *<!-- Пятница, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Физика</div>  
 <div class="lesson-room">43</div>  
 </div>  
   
 *<!-- Суббота, 1 пара -->* <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">МДК.01.01</div>  
 <div class="lesson-room">43</div>  
 </div>  
   
 *<!-- 2 ПАРА -->* <div class="schedule-cell lesson-number">2 пара</div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Английский язык</div>  
 <div class="lesson-room">42</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Русский язык</div>  
 <div class="lesson-room">13</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Информатика</div>  
 <div class="lesson-room">22</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Физкультура</div>  
 <div class="lesson-room">24</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Английский язык</div>  
 <div class="lesson-room">45</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Архитектура</div>  
 <div class="lesson-room">13</div>  
 </div>  
   
 *<!-- 3 ПАРА -->* <div class="schedule-cell lesson-number">3 пара</div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Математика</div>  
 <div class="lesson-room">12</div>  
 </div>  
   
 <div class="schedule-cell empty-cell">-</div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">История</div>  
 <div class="lesson-room">12</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Английский язык</div>  
 <div class="lesson-room">13</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">МДК.01.02</div>  
 <div class="lesson-room">43</div>  
 </div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">Математика</div>  
 <div class="lesson-room">22</div>  
 </div>  
   
 *<!-- 4 ПАРА -->* <div class="schedule-cell lesson-number">4 пара</div>  
   
 <div class="schedule-cell empty-cell">-</div>  
 <div class="schedule-cell empty-cell">-</div>  
   
 <div class="schedule-cell lesson-cell">  
 <div class="lesson-name">МДК.01.02</div>  
 <div class="lesson-room">32</div>  
 </div>  
   
 <div class="schedule-cell empty-cell">-</div>  
 <div class="schedule-cell empty-cell">-</div>  
 <div class="schedule-cell empty-cell">-</div>  
 </div>  
 </div>  
</section>

Листинг 32 – функции страницы расписания

// === РАСПИСАНИЕ ===  
async function loadGroupsForSchedule() {  
 const response = await fetch(`${API\_URL}/groups`);  
 groups = await response.json();  
  
 const select = document.getElementById('scheduleGroupSelect');  
 select.innerHTML = '<option value="">Выберите группу</option>' +  
 groups.map(g => `<option value="${g.id}">${g.name}</option>`).join('');  
}  
  
async function loadSchedule() {  
 const groupId = document.getElementById('scheduleGroupSelect').value;  
 if (!groupId) return;  
  
 const response = await fetch(`${API\_URL}/schedule?group\_id=${groupId}`);  
 const schedules = await response.json();  
  
 const subjectsResponse = await fetch(`${API\_URL}/subjects`);  
 subjects = await subjectsResponse.json();  
  
 const group = groups.find(g => g.id == groupId);  
 document.getElementById('scheduleTitle').innerHTML =  
 `<div class="schedule-title">Расписание для группы: ${group.name}</div>`;  
  
 const days = ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота'];  
 const tbody = document.getElementById('scheduleTable');  
 tbody.innerHTML = '';  
  
 for (let lesson = 1; lesson <= 4; lesson++) {  
 const tr = document.createElement('tr');  
 tr.innerHTML = `<td>${lesson} пара</td>`;  
  
 for (const day of days) {  
 const td = document.createElement('td');  
 const schedule = schedules.find(s => s.lesson\_number === lesson && s.day\_of\_week === day);  
  
 if (schedule) {  
 const subject = subjects.find(s => s.id === schedule.subject\_id);  
 td.innerHTML = `  
 <div class="lesson-card">  
 <div>${subject ? subject.name : 'Неизвестно'}</div>  
 ${schedule.room ? `<div class="room">${schedule.room}</div>` : ''}  
 </div>  
 `;  
 } else {  
 td.textContent = '-';  
 }  
  
 tr.appendChild(td);  
 }  
  
 tbody.appendChild(tr);  
 }  
}  
  
function exportSchedule() {  
 const groupId = document.getElementById('scheduleGroupSelect').value;  
 if (!groupId) {  
 alert('Выберите группу!');  
 return;  
 }  
 window.open(`${API\_URL}/export/schedule/${groupId}/excel`, '\_blank');  
}

Листинг 33 – CSS-стили

<style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
  
 body {  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
 background: #1a1d29;  
 color: #e4e4e7;  
 min-height: 100vh;  
 padding: 20px;  
 transition: background-color 0.3s, color 0.3s;  
 }  
  
 body.light-theme {  
 background: #f5f5f5;  
 color: #1a1d29;  
 }  
  
 .container {  
 max-width: 1400px;  
 margin: 0 auto;  
 }  
  
 .header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 20px;  
 }  
  
 .theme-toggle {  
 padding: 10px 20px;  
 border: none;  
 border-radius: 8px;  
 background: #2d3142;  
 color: #e4e4e7;  
 cursor: pointer;  
 font-size: 14px;  
 font-weight: 500;  
 transition: all 0.3s;  
 display: flex;  
 align-items: center;  
 gap: 8px;  
 }  
  
 .light-theme .theme-toggle {  
 background: #e0e0e0;  
 color: #1a1d29;  
 }  
  
 .theme-toggle:hover {  
 background: #3d4252;  
 }  
  
 .light-theme .theme-toggle:hover {  
 background: #d0d0d0;  
 }  
  
 .tabs {  
 display: flex;  
 gap: 15px;  
 justify-content: center;  
 margin-bottom: 30px;  
 }  
  
 .tab {  
 padding: 12px 30px;  
 border: none;  
 border-radius: 8px;  
 background: #2d3142;  
 color: #e4e4e7;  
 cursor: pointer;  
 font-size: 15px;  
 font-weight: 500;  
 transition: all 0.3s;  
 }  
  
 .light-theme .tab {  
 background: #e0e0e0;  
 color: #1a1d29;  
 }  
  
 .tab:hover {  
 background: #3d4252;  
 }  
  
 .light-theme .tab:hover {  
 background: #d0d0d0;  
 }  
  
 .tab.active {  
 background: #5B8DEF;  
 color: white;  
 }  
  
 .content-box {  
 background: #2d3142;  
 border-radius: 12px;  
 padding: 30px;  
 margin-bottom: 20px;  
 transition: background-color 0.3s;  
 }  
  
 .light-theme .content-box {  
 background: #ffffff;  
 box-shadow: 0 2px 8px rgba(0,0,0,0.1);  
 }  
  
 .tab-content {  
 display: none;  
 }  
  
 .tab-content.active {  
 display: block;  
 animation: fadeIn 0.3s;  
 }  
  
 @keyframes fadeIn {  
 from { opacity: 0; }  
 to { opacity: 1; }  
 }  
  
 .two-column {  
 display: grid;  
 grid-template-columns: 480px 1fr;  
 gap: 30px;  
 }  
  
 h2 {  
 font-size: 22px;  
 margin-bottom: 25px;  
 color: #e4e4e7;  
 }  
  
 .light-theme h2 {  
 color: #1a1d29;  
 }  
  
 .form-group {  
 margin-bottom: 20px;  
 }  
  
 .form-group label {  
 display: block;  
 margin-bottom: 8px;  
 font-size: 14px;  
 color: #a0a0a8;  
 }  
  
 .light-theme .form-group label {  
 color: #666666;  
 }  
  
 .form-group input,  
 .form-group select {  
 width: 100%;  
 padding: 12px 15px;  
 border: none;  
 border-radius: 8px;  
 background: #1a1d29;  
 color: #e4e4e7;  
 font-size: 14px;  
 transition: all 0.3s;  
 }  
  
 .light-theme .form-group input,  
 .light-theme .form-group select {  
 background: #f5f5f5;  
 color: #1a1d29;  
 border: 1px solid #e0e0e0;  
 }  
  
 .form-group input:focus,  
 .form-group select:focus {  
 outline: none;  
 background: #252837;  
 }  
  
 .light-theme .form-group input:focus,  
 .light-theme .form-group select:focus {  
 background: #ffffff;  
 border-color: #5B8DEF;  
 }  
  
 .form-group input::placeholder {  
 color: #5a5a68;  
 }  
  
 .light-theme .form-group input::placeholder {  
 color: #999999;  
 }  
  
 .btn {  
 padding: 12px 24px;  
 border: none;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 14px;  
 font-weight: 500;  
 transition: all 0.3s;  
 margin-right: 10px;  
 display: inline-flex;  
 align-items: center;  
 gap: 8px;  
 }  
  
 .btn-primary {  
 background: #5B8DEF;  
 color: white;  
 }  
  
 .btn-primary:hover {  
 background: #4a7dd9;  
 }  
  
 .btn-secondary {  
 background: #3d4252;  
 color: #e4e4e7;  
 }  
  
 .light-theme .btn-secondary {  
 background: #e0e0e0;  
 color: #1a1d29;  
 }  
  
 .btn-secondary:hover {  
 background: #4d5262;  
 }  
  
 .light-theme .btn-secondary:hover {  
 background: #d0d0d0;  
 }  
  
 .btn-success {  
 background: #10b981;  
 color: white;  
 font-size: 13px;  
 padding: 8px 16px;  
 }  
  
 .btn-success:hover {  
 background: #059669;  
 }  
  
 .btn-warning {  
 background: #f59e0b;  
 color: white;  
 }  
  
 .btn-warning:hover {  
 background: #d97706;  
 }  
  
 .btn-info {  
 background: #10b981;  
 color: white;  
 font-size: 13px;  
 padding: 8px 16px;  
 }  
  
 .btn-info:hover {  
 background: #059669;  
 }  
  
 .search-box {  
 margin-bottom: 20px;  
 }  
  
 .search-box input {  
 width: 100%;  
 padding: 12px 15px;  
 border: none;  
 border-radius: 8px;  
 background: #1a1d29;  
 color: #e4e4e7;  
 font-size: 14px;  
 }  
  
 .light-theme .search-box input {  
 background: #f5f5f5;  
 color: #1a1d29;  
 border: 1px solid #e0e0e0;  
 }  
  
 .student-card {  
 background: #1a1d29;  
 border-radius: 10px;  
 padding: 20px;  
 margin-bottom: 15px;  
 border: 1px solid #3d4252;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
  
 .light-theme .student-card {  
 background: #f5f5f5;  
 border-color: #e0e0e0;  
 }  
  
 .student-card:hover {  
 background: #252837;  
 border-color: #5B8DEF;  
 transform: translateY(-2px);  
 }  
  
 .light-theme .student-card:hover {  
 background: #ffffff;  
 border-color: #5B8DEF;  
 }  
  
 .student-card.selected {  
 border-color: #5B8DEF;  
 background: #252837;  
 }  
  
 .light-theme .student-card.selected {  
 background: #ffffff;  
 }  
  
 .student-card h3 {  
 font-size: 18px;  
 margin-bottom: 10px;  
 color: #e4e4e7;  
 }  
  
 .light-theme .student-card h3 {  
 color: #1a1d29;  
 }  
  
 .student-card p {  
 font-size: 14px;  
 color: #a0a0a8;  
 margin-bottom: 5px;  
 }  
  
 .light-theme .student-card p {  
 color: #666666;  
 }  
  
 .student-card .actions {  
 margin-top: 15px;  
 display: flex;  
 gap: 8px;  
 }  
  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 }  
  
 th {  
 background: #1a1d29;  
 color: #e4e4e7;  
 padding: 15px;  
 text-align: left;  
 font-weight: 600;  
 font-size: 14px;  
 }  
  
 .light-theme th {  
 background: #f5f5f5;  
 color: #1a1d29;  
 }  
  
 td {  
 padding: 15px;  
 border-top: 1px solid #3d4252;  
 color: #e4e4e7;  
 }  
  
 .light-theme td {  
 border-top: 1px solid #e0e0e0;  
 color: #1a1d29;  
 }  
  
 tr {  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
  
 tr:hover {  
 background: #252837;  
 }  
  
 .light-theme tr:hover {  
 background: #f5f5f5;  
 }  
  
 tr.selected {  
 background: #252837;  
 border-left: 3px solid #5B8DEF;  
 }  
  
 .light-theme tr.selected {  
 background: #f5f5f5;  
 }  
  
 .schedule-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 20px;  
 }  
  
 .schedule-table {  
 background: #1a1d29;  
 border-radius: 10px;  
 overflow: hidden;  
 }  
  
 .light-theme .schedule-table {  
 background: #ffffff;  
 border: 1px solid #e0e0e0;  
 }  
  
 .schedule-table table {  
 margin: 0;  
 }  
  
 .schedule-table th {  
 background: #2d3142;  
 padding: 12px;  
 font-size: 13px;  
 text-align: center;  
 }  
  
 .light-theme .schedule-table th {  
 background: #e0e0e0;  
 }  
  
 .schedule-table td {  
 padding: 20px 10px;  
 text-align: center;  
 vertical-align: middle;  
 }  
  
 .lesson-card {  
 background: #5B8DEF;  
 color: white;  
 padding: 10px;  
 border-radius: 6px;  
 font-size: 13px;  
 font-weight: 500;  
 }  
  
 .lesson-card .room {  
 font-size: 11px;  
 opacity: 0.9;  
 margin-top: 4px;  
 }  
  
 .export-btn {  
 background: #10b981;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 14px;  
 display: inline-flex;  
 align-items: center;  
 gap: 8px;  
 }  
  
 .export-btn:hover {  
 background: #059669;  
 }  
  
 .schedule-title {  
 background: #5B8DEF;  
 color: white;  
 padding: 15px;  
 border-radius: 8px;  
 text-align: center;  
 font-size: 16px;  
 font-weight: 600;  
 margin-bottom: 20px;  
 }  
  
 .schedule-header select {  
 padding: 10px 15px;  
 background: #1a1d29;  
 color: #e4e4e7;  
 border: none;  
 border-radius: 8px;  
 font-size: 14px;  
 }  
  
 .light-theme .schedule-header select {  
 background: #f5f5f5;  
 color: #1a1d29;  
 border: 1px solid #e0e0e0;  
 }  
  
 .schedule-header label {  
 margin-right: 10px;  
 color: #a0a0a8;  
 }  
  
 .light-theme .schedule-header label {  
 color: #666666;  
 }

## 4.3. Реализация модулей вывода на печать

- вывод в шаблон .docx

Для создания документов Word используется библиотека **python-docx.** Само создание word-файла делается шаблонно в модуле export\_utils.py, функция create\_student\_certificate()

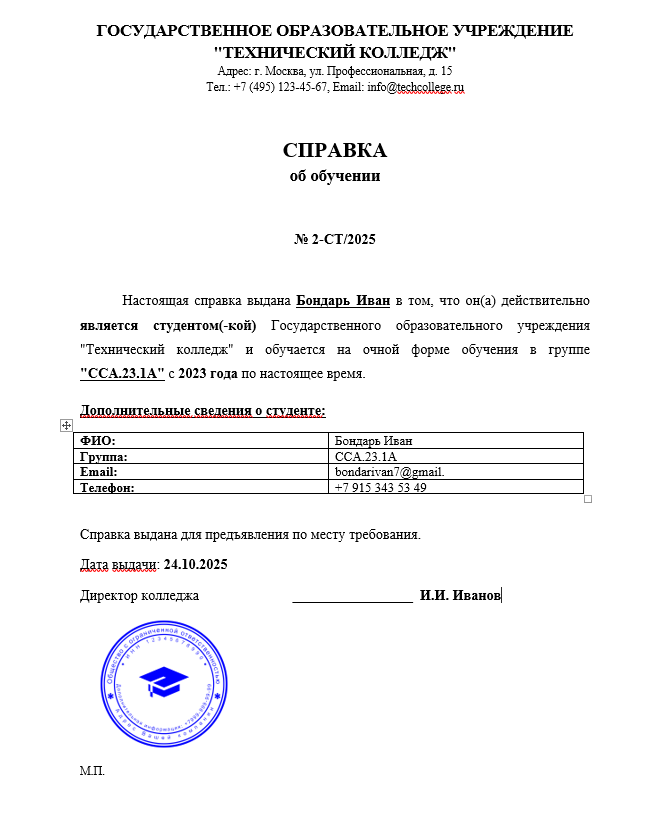


Рис. 15 «Шаблон docx»

Листинг 34 – создание docx-файла

def create\_student\_certificate(student, group\_name):  
 *"""Создание справки об обучении студента из Word-шаблона"""  
  
 # Путь к шаблону* template\_path = 'certificate\_template.docx'  
  
 *# ЗАГРУЖАЕМ ШАБЛОН (НЕ СОЗДАЕМ НОВЫЙ ДОКУМЕНТ!)* doc = Document(template\_path)  
  
 *# Подготовка данных для замены плейсхолдеров* replacements = {  
 '{{ref\_number}}': f'{student.id}-СТ/{datetime.now().year}',  
 '{{student\_fio}}': f'{student.surname} {student.name}',  
 '{{group\_name}}': group\_name or 'Не указана',  
 '{{study\_year}}': str(datetime.now().year - 2),  
 '{{email}}': student.email or 'Не указан',  
 '{{phone}}': student.phone or 'Не указан',  
 '{{issue\_date}}': datetime.now().strftime('%d.%m.%Y')  
 }  
  
 *# Замена плейсхолдеров в параграфах* for paragraph in doc.paragraphs:  
 for key, value in replacements.items():  
 if key in paragraph.text:  
 for run in paragraph.runs:  
 if key in run.text:  
 run.text = run.text.replace(key, value)  
  
 *# Замена плейсхолдеров в таблицах* for table in doc.tables:  
 for row in table.rows:  
 for cell in row.cells:  
 for paragraph in cell.paragraphs:  
 for key, value in replacements.items():  
 if key in paragraph.text:  
 for run in paragraph.runs:  
 if key in run.text:  
 run.text = run.text.replace(key, value)  
  
 *# === ДОБАВЛЕНИЕ ПЕЧАТИ ===  
 # Ищем параграф с текстом "М.П." или добавляем печать в конец документа* stamp\_path = 'stamp.png' *# Путь к файлу печати* if os.path.exists(stamp\_path):last\_paragraph = doc.paragraphs[-1]  
  
 *# Создаем новый параграф для печати* stamp\_paragraph = last\_paragraph.insert\_paragraph\_before()  
 stamp\_run = stamp\_paragraph.add\_run()  
  
 *# Вставляем изображение печати  
 # Размер: 4см x 4см (компактная печать)* stamp\_run.add\_picture(stamp\_path, width=Cm(4), height=Cm(4))  
  
 *# Выравнивание по левому краю (где обычно ставится М.П.)* stamp\_paragraph.alignment = WD\_ALIGN\_PARAGRAPH.LEFT  
  
 *# Добавляем отступ слева для выравнивания с текстом "М.П."* stamp\_paragraph.paragraph\_format.left\_indent = Cm(0.5)  
  
 *# Сохранение в буфер* buffer = io.BytesIO()  
 doc.save(buffer)  
 buffer.seek(0)  
  
 return buffer

Листинг 35 – API для экспорта docx-файла

@app.get("/api/export/student/{student\_id}/certificate-word")  
def export\_student\_certificate\_word(student\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт справки студента в Word"""* student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
  
 group\_name = None  
 if student.group\_id:  
 group = db.query(models.Group).filter(models.Group.id == student.group\_id).first()  
 group\_name = group.name if group else None  
  
 buffer = create\_student\_certificate(student, group\_name)  
  
 *# Безопасное имя файла без русских символов* filename = f"certificate\_{student.surname}\_{student.name}.docx"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/vnd.openxmlformats-officedocument.wordprocessingml.document",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

- создание документа .xlsx

Для работы с excel используется библиотека **openpyxl.** Само создание excel-файла делается в модуле export\_utils.py, функция create\_schelude\_excel()



Рис. 16 «Пример excel»

Листинг 36 – создание excel-файла

def create\_schedule\_excel(group, schedules, subjects\_dict):  
 *"""Создание расписания в Excel с профессиональным оформлением"""* wb = Workbook()  
 ws = wb.active  
 ws.title = "Расписание"  
  
 *# === СТИЛИ ===* header\_font = Font(name='Arial', size=14, bold=True, color='FFFFFF')  
 header\_fill = PatternFill(start\_color='4472C4', end\_color='4472C4', fill\_type='solid')  
 cell\_font = Font(name='Arial', size=11)  
 title\_font = Font(name='Arial', size=16, bold=True)  
 subtitle\_font = Font(name='Arial', size=12, bold=True)  
  
 border = Border(  
 left=Side(style='thin'),  
 right=Side(style='thin'),  
 top=Side(style='thin'),  
 bottom=Side(style='thin')  
 )  
  
 *# === ШАПКА ДОКУМЕНТА ===* ws.merge\_cells('A1:G1')  
 cell = ws['A1']  
 cell.value = 'ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ'  
 cell.font = title\_font  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
 ws.row\_dimensions[1].height = 25  
  
 ws.merge\_cells('A2:G2')  
 cell = ws['A2']  
 cell.value = '"ТЕХНИЧЕСКИЙ КОЛЛЕДЖ"'  
 cell.font = Font(name='Arial', size=14, bold=True)  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
 ws.row\_dimensions[2].height = 20  
  
 ws.merge\_cells('A3:G3')  
 cell = ws['A3']  
 cell.value = 'г. Москва, ул. Профессиональная, д. 15'  
 cell.font = Font(name='Arial', size=10)  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
  
 ws.merge\_cells('A5:G5')  
 cell = ws['A5']  
 cell.value = f'РАСПИСАНИЕ ЗАНЯТИЙ'  
 cell.font = Font(name='Arial', size=14, bold=True)  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
 ws.row\_dimensions[5].height = 25  
  
 ws.merge\_cells('A6:G6')  
 cell = ws['A6']  
 cell.value = f'Группа: {group.name}'  
 cell.font = subtitle\_font  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
 ws.row\_dimensions[6].height = 20  
  
 ws.merge\_cells('A7:G7')  
 cell = ws['A7']  
 cell.value = f'Учебный год: {datetime.now().year}-{datetime.now().year + 1}'  
 cell.font = Font(name='Arial', size=10, italic=True)  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
  
 *# === ЗАГОЛОВКИ ТАБЛИЦЫ ===* days = ['Пара', 'Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота']  
 row = 9  
  
 for col, day in enumerate(days, start=1):  
 cell = ws.cell(row=row, column=col)  
 cell.value = day  
 cell.font = header\_font  
 cell.fill = header\_fill  
 cell.alignment = Alignment(horizontal='center', vertical='center')  
 cell.border = border  
  
 ws.row\_dimensions[row].height = 30  
  
 *# === НАСТРОЙКА ШИРИНЫ КОЛОНОК ===* ws.column\_dimensions['A'].width = 10  
 for col in ['B', 'C', 'D', 'E', 'F', 'G']:  
 ws.column\_dimensions[col].width = 22  
  
 *# === ЗАПОЛНЕНИЕ РАСПИСАНИЯ ===* lesson\_times = [  
 '08:30 - 10:00',  
 '10:10 - 11:40',  
 '12:00 - 13:30',  
 '13:40 - 15:10'  
 ]  
  
 for lesson\_num in range(1, 5):  
 row += 1  
  
 *# Номер пары с временем* cell = ws.cell(row=row, column=1)  
 cell.value = f'{lesson\_num}\n{lesson\_times[lesson\_num - 1]}'  
 cell.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
 cell.border = border  
 cell.font = Font(name='Arial', size=10, bold=True)  
  
 for col, day in enumerate(days[1:], start=2):  
 cell = ws.cell(row=row, column=col)  
  
 *# Поиск занятия* schedule\_item = next((s for s in schedules  
 if s.lesson\_number == lesson\_num and s.day\_of\_week == day), None)  
  
 if schedule\_item:  
 subject\_name = subjects\_dict.get(schedule\_item.subject\_id, 'Неизвестно')  
 room = f'\n(ауд. {schedule\_item.room})' if schedule\_item.room else ''  
 cell.value = f'{subject\_name}{room}'  
 cell.fill = PatternFill(start\_color='E7E6E6', end\_color='E7E6E6', fill\_type='solid')  
 else:  
 cell.value = '-'  
  
 cell.alignment = Alignment(horizontal='center', vertical='center', wrap\_text=True)  
 cell.border = border  
 cell.font = cell\_font  
  
 ws.row\_dimensions[row].height = 50  
  
 *# === ПОДПИСЬ И ДАТА ===* row += 3  
 ws.merge\_cells(f'A{row}:G{row}')  
 cell = ws[f'A{row}']  
 cell.value = f'Дата формирования расписания: {datetime.now().strftime("%d.%m.%Y")}'  
 cell.font = Font(name='Arial', size=10, italic=True)  
 cell.alignment = Alignment(horizontal='center')  
  
 row += 2  
 ws.merge\_cells(f'A{row}:D{row}')  
 cell = ws[f'A{row}']  
 cell.value = 'Заместитель директора по УР:'  
 cell.font = Font(name='Arial', size=11)  
  
 ws.merge\_cells(f'E{row}:F{row}')  
 cell = ws[f'E{row}']  
 cell.value = '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 cell.font = Font(name='Arial', size=11)  
 cell.alignment = Alignment(horizontal='center')  
  
 cell = ws[f'G{row}']  
 cell.value = 'С.П. Сидоров'  
 cell.font = Font(name='Arial', size=11, bold=True)  
  
 *# Сохранение в буфер* buffer = io.BytesIO()  
 wb.save(buffer)  
 buffer.seek(0)  
  
 return buffer

Листинг 37 – API для экспорта excel-файла

@app.get("/api/export/schedule/{group\_id}/excel")  
def export\_schedule\_excel(group\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт расписания группы в Excel"""* group = db.query(models.Group).filter(models.Group.id == group\_id).first()  
 if not group:  
 raise HTTPException(status\_code=404, detail="Группа не найдена")  
  
 schedules = db.query(models.Schedule).filter(models.Schedule.group\_id == group\_id).all()  
 subjects = db.query(models.Subject).all()  
 subjects\_dict = {s.id: s.name for s in subjects}  
  
 buffer = create\_schedule\_excel(group, schedules, subjects\_dict)  
  
 *# Безопасное имя файла без русских символов* filename = f"schedule\_{group.name}.xlsx"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

- создание документа .pdf

Для генерации pdf-файла используется библиотека ReportLab. Сама генерация принимает форму листа А4 с красивой рамкой и вставленной информацией с помощью модуля export\_utils.py, функция create\_student\_certificate\_pdf().



Рис. 17 «Пример pdf»

Листинг 38 – создание pdf-файла

def create\_student\_certificate\_pdf(student, group\_name):  
 *"""Создание справки студента в PDF - отличается от Word версии"""* buffer = io.BytesIO()  
 c = canvas.Canvas(buffer, pagesize=A4)  
 width, height = A4  
  
 *# Регистрация русского шрифта (DejaVu Sans поддерживает кириллицу)* try:  
 *# Пробуем загрузить DejaVu Sans* pdfmetrics.registerFont(TTFont('DejaVu', '/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf'))  
 pdfmetrics.registerFont(TTFont('DejaVu-Bold', '/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf'))  
 font\_regular = 'DejaVu'  
 font\_bold = 'DejaVu-Bold'  
 except:  
 *# Если не получилось, пробуем Arial (Windows)* try:  
 pdfmetrics.registerFont(TTFont('Arial', 'arial.ttf'))  
 pdfmetrics.registerFont(TTFont('Arial-Bold', 'arialbd.ttf'))  
 font\_regular = 'Arial'  
 font\_bold = 'Arial-Bold'  
 except:  
 *# В крайнем случае используем встроенный Helvetica (но он не поддерживает кириллицу)* font\_regular = 'Helvetica'  
 font\_bold = 'Helvetica-Bold'  
  
 *# === РАМКА ДОКУМЕНТА ===* c.setStrokeColorRGB(0.2, 0.2, 0.8)  
 c.setLineWidth(2)  
 c.rect(40, 40, width - 80, height - 80, stroke=1, fill=0)  
  
 *# === ШАПКА ===* c.setFont(font\_bold, 16)  
 c.drawCentredString(width / 2, height - 80, "МИНИСТЕРСТВО ОБРАЗОВАНИЯ")  
  
 c.setFont(font\_bold, 14)  
 c.drawCentredString(width / 2, height - 105, "ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ")  
 c.drawCentredString(width / 2, height - 125, '"ТЕХНИЧЕСКИЙ КОЛЛЕДЖ"')  
  
 *# Линия разделителя* c.setStrokeColorRGB(0.5, 0.5, 0.5)  
 c.setLineWidth(1)  
 c.line(80, height - 140, width - 80, height - 140)  
  
 *# === ЗАГОЛОВОК ===* c.setFont(font\_bold, 20)  
 c.drawCentredString(width / 2, height - 180, "СПРАВКА")  
  
 c.setFont(font\_bold, 12)  
 ref\_number = f"№ {student.id}-PDF/{datetime.now().year}"  
 c.drawCentredString(width / 2, height - 205, ref\_number)  
  
 *# === ОСНОВНОЙ ТЕКСТ ===* c.setFont(font\_regular, 12)  
 y = height - 250  
  
 *# Многострочный текст с переносами* text\_width = width - 160  
  
 text1 = f"Выдана студенту(ке) {student.surname} {student.name}"  
 c.drawString(80, y, text1)  
 y -= 25  
  
 text2 = "в том, что он(а) обучается в"  
 c.drawString(80, y, text2)  
 y -= 20  
  
 c.setFont(font\_bold, 12)  
 text3 = 'Государственном образовательном учреждении'  
 c.drawString(80, y, text3)  
 y -= 20  
  
 text4 = '"Технический колледж"'  
 c.drawString(80, y, text4)  
 y -= 25  
  
 c.setFont(font\_regular, 12)  
 text5 = f"по программе среднего профессионального образования"  
 c.drawString(80, y, text5)  
 y -= 20  
  
 text6 = f"в группе: {group\_name or 'Не указана'}"  
 c.drawString(80, y, text6)  
 y -= 20  
  
 text7 = f"с {datetime.now().year - 2} года по настоящее время."  
 c.drawString(80, y, text7)  
  
 *# === ТАБЛИЦА С ДАННЫМИ ===* y -= 40  
 c.setFont(font\_bold, 11)  
 c.drawString(80, y, "Контактные данные студента:")  
 y -= 25  
  
 *# Рисуем таблицу* c.setStrokeColorRGB(0.3, 0.3, 0.3)  
 c.setLineWidth(0.5)  
  
 *# Заголовки* c.setFont(font\_bold, 10)  
 c.drawString(90, y, "ФИО:")  
 c.drawString(90, y - 20, "Группа:")  
 c.drawString(90, y - 40, "Email:")  
 c.drawString(90, y - 60, "Телефон:")  
  
 *# Значения* c.setFont(font\_regular, 10)  
 c.drawString(200, y, f"{student.surname} {student.name}")  
 c.drawString(200, y - 20, f"{group\_name or 'Не указана'}")  
 c.drawString(200, y - 40, f"{student.email or 'Не указан'}")  
 c.drawString(200, y - 60, f"{student.phone or 'Не указан'}")  
  
 *# Рамка таблицы* c.rect(85, y - 75, width - 170, 90, stroke=1, fill=0)  
  
 *# === НАЗНАЧЕНИЕ ===* y -= 100  
 c.setFont(font\_regular, 11)  
 c.drawString(80, y, "Справка дана для предъявления по месту требования.")  
  
 *# === ДАТА ===* y -= 40  
 c.setFont(font\_bold, 11)  
 date\_str = f"Дата выдачи: {datetime.now().strftime('%d.%m.%Y')}"  
 c.drawString(80, y, date\_str)  
  
 *# === ПОДПИСИ ===* y -= 60  
 c.setFont(font\_regular, 11)  
 c.drawString(80, y, "Директор колледжа")  
 c.drawString(270, y, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
 c.setFont(font\_bold, 11)  
 c.drawString(420, y, "И.И. Иванов")  
  
 y -= 30  
 c.setFont(font\_regular, 11)  
 c.drawString(80, y, "Зам. директора по УР")  
 c.drawString(270, y, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
 c.setFont(font\_bold, 11)  
 c.drawString(420, y, "С.П. Сидоров")  
  
 *# === ПЕЧАТЬ ===* y -= 40  
 c.setFont(font\_regular, 10)  
 c.drawString(80, y, "М.П.")  
  
 *# Добавляем изображение печати* stamp\_path = 'stamp.png'  
 if os.path.exists(stamp\_path):  
 *# Вставляем печать рядом с "М.П."* c.drawImage(stamp\_path, 110, y - 50, width=80, height=80, preserveAspectRatio=True, mask='auto')  
  
 c.save()  
 buffer.seek(0)  
  
 return buffer

Листинг 39 – API для экспорт pdf-файла

@app.get("/api/export/student/{student\_id}/certificate-pdf")  
def export\_student\_certificate\_pdf(student\_id: int, db: Session = Depends(get\_db)):  
 *"""Экспорт справки студента в PDF"""* student = db.query(models.Student).filter(models.Student.id == student\_id).first()  
 if not student:  
 raise HTTPException(status\_code=404, detail="Студент не найден")  
  
 group\_name = None  
 if student.group\_id:  
 group = db.query(models.Group).filter(models.Group.id == student.group\_id).first()  
 group\_name = group.name if group else None  
  
 buffer = create\_student\_certificate\_pdf(student, group\_name)  
  
 *# Безопасное имя файла без русских символов* filename = f"certificate\_{student.surname}\_{student.name}.pdf"  
 safe\_filename = urllib.parse.quote(filename)  
  
 return StreamingResponse(  
 buffer,  
 media\_type="application/pdf",  
 headers={  
 "Content-Disposition": f"attachment; filename={safe\_filename}; filename\*=UTF-8''{safe\_filename}"  
 }  
 )

## 4.4. Реализация импорта данных из файла в базу данных

## 4.5. Загрузка приложения в репозиторий GitHub

# 5. Тестирование и отладка

## 5.1. Проведение модульного тестирования

## 5.2. Отладка программы

# 6. Разработка мобильной версии

# 7. Разработка технической документации

## 7.1. Руководство пользователя

## 7.2. Руководство программиста

## 7.3. Руководство системного программиста

## 7.4. Методика тестирования и испытания программы

# Заключение

# Список литературы