Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Алтайский государственный технический университет
им. И.И. Ползунова»

Факультет __информационных технологий_____
Кафедра прикладной математики
Специальность (направление, профиль)__ПИ_____

Курсовой проект
защищен с
оценкой 70 (семьдесят)
_____М.А.Кайгородова
(подпись руководителя проекта) (инициалы, фамилия)
"29" 12 2025 г.

КУРСОВОЙ ПРОЕКТ

___Разработка информационной системы «Список школ Алтайского края»___
(тема курсового проекта)

Пояснительная записка

по дисциплине _____Курсовая работа по базам данных_____

_____КП 09.03.04.33.000 ПЗ_____
(обозначение документа)

Студент группы ПИ-21 Лихтинфельд А.А._____26.12.2025
(фамилия, имя, отчество) (подпись) (дата)

Студент группы ПИ-21 Подгорный И.В._____26.12.2025
(фамилия, имя, отчество) (подпись) (дата)

Доцент, к.т.н._____М.А.Кайгородова
(должность, ученое звание) (подпись) (инициалы, фамилия)

БАРНАУЛ 2025

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Алтайский государственный технический университет
имени И.И. Ползунова»
Факультет информационных технологий
Кафедра «Прикладная математика»

З А Д А Н И Е

на курсовой проект по дисциплине «Курсовая работа по базам данных»

студентам группы ПИ-21
Лихтинфельд Анне Алексеевне, Подгорному Ивану Валерьевичу

Тема курсового проекта: “Разработка информационной системы «Список школ Алтайского края»”.

Календарный план работы:

№ этапа	Содержание этапа	Недели семестра
1	Получение задания	1
2	Постановка задачи. Работа с документацией	2
3	Проектирование программы	5
4	Реализация программы	14
5	Оформление пояснительной записки	15
6	Защита курсового проекта	15-16

Руководитель проекта_____ Кайгородова М.А., доцент, к.т.н.

подпись

Задание принял к исполнению _____Лихтинфельд А.А.

подпись

Задание принял к исполнению _____Подгорный И.В.

подпись

Аннотация

Курсовой проект по дисциплине «Курсовая работа по базам данных» на тему «Разработка информационной системы «Список школ Алтайского края» состоит из описания предметной области, спроектированных концептуальной, логической и физической моделей, описания разработанного программного обеспечения по работе со списком школ Алтайского края, заключения, списка литературы, а также приложений, в которых представлено: тестирование программного продукта, скрипт для создания базы данных, код исходного приложения.

Содержание

Введение

Одной из важнейших задач, решаемых современными информационными системами, является централизованное управление данными, обеспечение их целостности, актуальности и удобного доступа для различных категорий пользователей.

В сфере образования Алтайского края существует объективная необходимость в систематизации информации об учебных заведениях. Регион объединяет более 1500 городских и сельских образовательных учреждений различного типа: общеобразовательные школы, гимназии, лицеи, школы-интернаты и коррекционные школы. Учёт таких организаций, ведение актуальных данных об их инфраструктуре, специализации, кадрах и программах представляет собой сложную задачу. В настоящий момент сбор и обновление информации часто ведутся вручную, что приводит к несвоевременному обновлению данных, их разрозненности и отсутствию единой точки доступа.

Таким образом, проблема автоматизации процесса сбора, хранения и предоставления данных об образовательных учреждениях становится крайне актуальной. Необходима система, которая позволит не только эффективно управлять этим массивом информации, но и предоставит удобный инструмент для родителей при выборе школы, для органов власти – при анализе и планировании, а для педагогов и учеников – для поиска образовательных возможностей.

Объектом исследования являются бизнес-процессы сбора, обработки и предоставления информации об образовательных учреждениях на территории Алтайского края.

Предметом исследования выступает проектируемая информационная система «Реестр школ Алтайского края».

Целью данной работы является проектирование и разработка веб-приложения для комплексного управления данными об образовательных учреждениях региона.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать предметную область и сформулировать требования к системе.
2. Спроектировать структуру базы данных, обеспечивающую хранение всей необходимой информации.
3. Разработать серверную часть приложения (backend) с реализацией бизнес-логики и API.
4. Создать интуитивно понятный и адаптивный пользовательский интерфейс (frontend).
5. Реализовать ключевой функционал: ведение реестра, расширенный поиск с фильтрами, формирование отчётов, модуль отзывов и журнал аудита изменений.
6. Обеспечить безопасность данных и соответствие требованиям законодательства (в частности, Федеральному закону № 273-ФЗ «Об образовании в РФ»).

1. Обзор предметной области и постановка задачи

1.1. Обзор предметной области

Алтайский край – субъект Российской Федерации, объединяющий городские и сельские территории с развитой образовательной инфраструктурой. В регионе ведется учет образовательных учреждений, включая общеобразовательные школы, гимназии, лицеи, школы-интернаты и коррекционные школы. Каждое учреждение характеризуется набором параметров: официальное название, юридический адрес, контактные данные (телефон, электронная почта, сайт), информация о руководстве (ФИО директора), количество учащихся, год основания, а также сведения о лицензии и аккредитации. Дополнительно указывается специализация школы (например, углубленное изучение предметов, профильные классы) и инфраструктура (наличие спортзалов, лабораторий, библиотек).

Данные формируются на основе информации от региональных органов образования, официальных отчетов школ, федеральных реестров (например, портал госуслуг) и сведений Рособрнадзора. Важным аспектом является соответствие законодательству, включая закон № 273-ФЗ "Об образовании", требования к защите персональных данных и правила публикации информации об учреждениях.

Список школ может быть дополнен описанием образовательных программ (дополнительные курсы, международные проекты) и социальных инициатив, таких как инклюзивное образование.

Информация публикуется на официальных сайтах региона. Пользователями данных являются родители, выбирающие школу для детей, государственные органы, планирующие бюджет, а также учителя и ученики, ищущие образовательные возможности.

К текущим проблемам относятся неактуальность данных из-за ручного обновления, недостаток информации о сельских школах и отсутствие централизованной платформы. Перспективы развития включают автоматизацию мониторинга изменений, интеграцию с федеральными программами и добавление модуля отзывов.

1.2. Постановка задачи

Необходимо разработать приложение для работы со списком школ Алтайского края, спроектировав и реализовав серверную часть и web-интерфейс приложения.

Для этого разобьем главную задачу на несколько частей:

- создание концептуальной и реляционной базы данных на основе вышеописанной предметной области;
- разработать базу данных и реализовать web-интерфейс;
- описание программного продукта и требований к системе.

При проектировании необходимо учесть особенности протекания бизнес-процессов в школах, описанные выше.

1.3. Техническое задание

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На разработку информационной системы
«Реестр школ Алтайского края»

№ п/п	Перечень основных требований	Содержание требований
1. Общие сведения		
1.1	Полное наименование	Информационная система «Реестр образовательных учреждений Алтайского края»
1.2	Наименование организации-заказчика	Министерство образования и науки Алтайского края
1.3	Наименование организации-разработчика	ООО «ЭдуТех»
1.4.	Перечень документов	Требования Рособрнадзора к учету образовательных учреждений Федеральный закон № 273-ФЗ «Об образовании в РФ»
1.5	Плановые сроки начала работ	21.04.2025
1.6	Плановые сроки окончания работ	31.08.2025
1.7	Источник и порядок финансирования работ	Финансирование за счет средств регионального бюджета Алтайского края

2. Цели и назначение создания автоматизированной системы		
2.1	Цели создания АС	Автоматизация учета образовательных учреждений края. Централизация информации для повышения доступности и прозрачности.
2.2	Назначение АС	Управление данными об образовательных учреждениях, включая школы, гимназии, лицеи, интернаты, и предоставление информации родителям, государственным органам, учителям и учащимся.
3. Характеристика объектов автоматизации		
3.1	Основные сведения об объекте автоматизации	Объект автоматизации — 1500+ образовательных учреждений Алтайского края, включая городские и сельские школы, гимназии, лицеи, интернаты.
3.2	Сведения об условиях эксплуатации объекта автоматизации и характеристика	Система должна функционировать в веб-среде.
4. Требования к автоматизированной системе		
4.1	Требования к структуре	1. Управление данными (CRUD-операции). 2. Поиск и фильтрация. 3. Отчетность. 4. Модуль отзывов.
4.2	Требования к функциям	Фильтрация по типу школы, инфраструктуре, программам. Импорт данных из Excel/CSV. Генерация отчетов по количеству учащихся, оснащенности, кадрам. Автоматическая проверка корректности вводимвх данных.
4.3	Требования к видам обеспечения	База данных на PostgreSQL. Серверы с резервированием данных. SSL-шифрование, двухфакторная аутентификация для администраторов.
4.4	Общие технические требования к АС	Резервное копирование данных ежедневно. Адаптивный интерфейс для ПК и мобильных устройств
5. Состав и содержание работ по созданию автоматизированной системы		
5.1	Реализация интерфейса	21.04.2025-27.05.2025
5.2	Наложение функционала	31.05.2025-28.06.2025
5.3	Первый тестовый запуск	10.06.2025-29.07.2025
5.4	Внедрение правок	25.06.2025-10.08.2025
5.5	Второй тестовый запуск	11.07.2025-25.08.2025
5.6	Внедрение правок	26.08.2025-10.09.2025
5.7	Запуск проекта	31.09.2025
6. Порядок разработки автоматизированной системы		
6.1	Порядок организации разработки АС	Разработка системы предполагается по укрупненному календарному плану, приведенному в П5
6.2	Перечень документов и	Модели базы данных

	исходных данных для разработки АС	Отчёты школ Дизайн системы Техническое задание на разработку АС. Текущие бизнес-процессы. Информация о существующей ИТ-инфраструктуре. Правовые и регуляторные требования. Требования к информационной безопасности.
6.3	Перечень документов, предъявляемых по окончании соответствующих этапов работ	Проектная документация. Отчеты о тестировании и испытаниях. Инструкции для пользователей. Акты о внедрении АС. Протоколы приемки системы.
6.4	Порядок проведения экспертизы технической документации	1. Формирование экспертной комиссии. 2. Изучение и анализ технической документации. 3. Сверка документации с нормативными требованиями. 4. Подготовка заключения экспертизы. Анализ использования вычислительной техники в измерительных операциях.
6.5	Перечень макетов, порядок их разработки, изготовления, испытаний, необходимость разработки на них документации, программы и методик испытаний	Макеты: 1. Прототип интерфейса поиска школ. 2. Макет отчета по оснащенности. 3. Пилотная версия модуля отзывов. Порядок: 1. Согласование дизайна с заказчиком. 2. Разработка функциональных макетов. 3. Тестирование на фокус-группе (учителя, родители). 4. Доработка по результатам обратной связи.
6.6	Порядок разработки автоматизированной системы, согласования и утверждения плана совместных работ по разработке АС	Порядок разработки: 1. Подготовка: Определение целей АС, формирование команды, анализ текущих процессов 2. Проектирование: Составление ТЗ, выбор методологии, проектирование архитектуры, подготовка проектной документации. 3. Разработка: Кодирование и тестирование ПО, создание пользовательских инструкций, разработка плана внедрения. 4. Внедрение: Установка системы, обучение пользователей, миграция данных, техподдержка. 5. Эксплуатация: Мониторинг системы, обновление и улучшение, дополнительное обучение. 6. Завершение: Формальная приемка системы заказчиком, подготовка окончательной документации.

		Редактирование плана будет происходить только при переговорах с заказчиками АС
6.7	Порядок разработки, согласования и утверждения программы работ по стандартизации	Внедрение стандартов информационной безопасности (ISO 27001)
6.8	Требования к гарантийным обязательствам разработчика	Гарантия 12 месяцев на устранение критических ошибок. Условия гарантии: – Сбой воспроизводится на резервной копии системы. – Отказ не вызван действиями пользователя. – Претензия соответствует требованиям ТЗ.
6.9	Порядок контроля и приемки автоматизированной АС	Испытания: 1. Проверка фильтров (например, «гимназии в Бийске»). 2. Тестирование импорта данных из CSV. 3. Аудит безопасности (проверка утечек данных) Приемка: – Подписание акта комиссией Минобразования края.
6.10	Порядок разработки, согласования и утверждения программы метрологического обеспечения, программы обеспечения надежности, программы эргономического обеспечения	Метрологическое обеспечение: – Валидация данных (проверка корректности вводимых адресов, контактов). Надежность: – Резервирование серверов. – Ежедневное бэкапирование БД. Эргономика: – Адаптивный дизайн для мобильных устройств. – Тестирование юзабилити с участием родителей.
7. Порядок контроля и приемки автоматизированной системы		
7.1	Виды, состав и методы испытаний	Испытание работоспособности АС будет проверяться на практике, путем внедрения в работу дирекции
7.2	Общие требования к приемке работ, порядок согласования и утверждения приемочной документации	Общие требования: 1. Соответствие выполненных работ техническому заданию. 2. Полная функциональность и безопасность АС. 3. Предоставление отчетов о тестировании и испытаниях. Порядок согласования: 1. Предъявление результатов работ заказчику. 2. Проверка и согласование документации заказчиком.

		3. Утверждение документации и подписание актов приемки. АС должна выполнять цели и назначения, для которых она создана, иметь интуитивно понятный интерфейс и выдавать оформленные по ГОСТу документы
7.3	Статус приемочной комиссии	Комиссия под руководством Министерства образования Алтайского края.
8. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу автоматизированной системы в действие		
8.1	Создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой АС требованиям ТЗ	Обеспечение инфраструктуры и ресурсов для работы АС в соответствии с ТЗ. При передаче Заказчику АС для эксплуатации, должна быть передана инструкция по эксплуатации АС
8.2	Проведение необходимых организационно-штатных мероприятий	Заказчик должен обеспечивать проведение опытной эксплуатации АС на реальном объеме данных при тестовых запусках
8.3	Порядок обучения персонала и пользователей АС	Период ознакомления с АС будет проходить во время тестовых запусков. В этот период можно будет обращаться напрямую к разработчикам по вопросам эксплуатации системы
9. Требования к документированию		
9.1	Перечень подлежащих разработке документов	Устав проекта, пользовательское соглашение, инструкция по эксплуатации АС
9.2	Вид представления и количество документов	Электронные и бумажные копии Макеты документов будут разработаны в соответствии с требованиями и прикреплены к ТЗ
9.3	Требования по использованию ЕСКД и ЕСПД при разработке документов	На основе ЕСКД и ЕСПД допускается, при необходимости, разрабатывать стандарты, учитывающие особенности выполнения документов
10. Источники разработки		
10	Документы и информационные материалы	Отчетные материалы должны включать в себя текстовые материалы (представленные в виде бумажной копии и на цифровом носителе в формате MS Word) и графические материалы

2. Проектирование

2.1. Концептуальная модель

При проектировании была разработана концептуальная модель (рис.1) согласно описанной ранее предметной области.

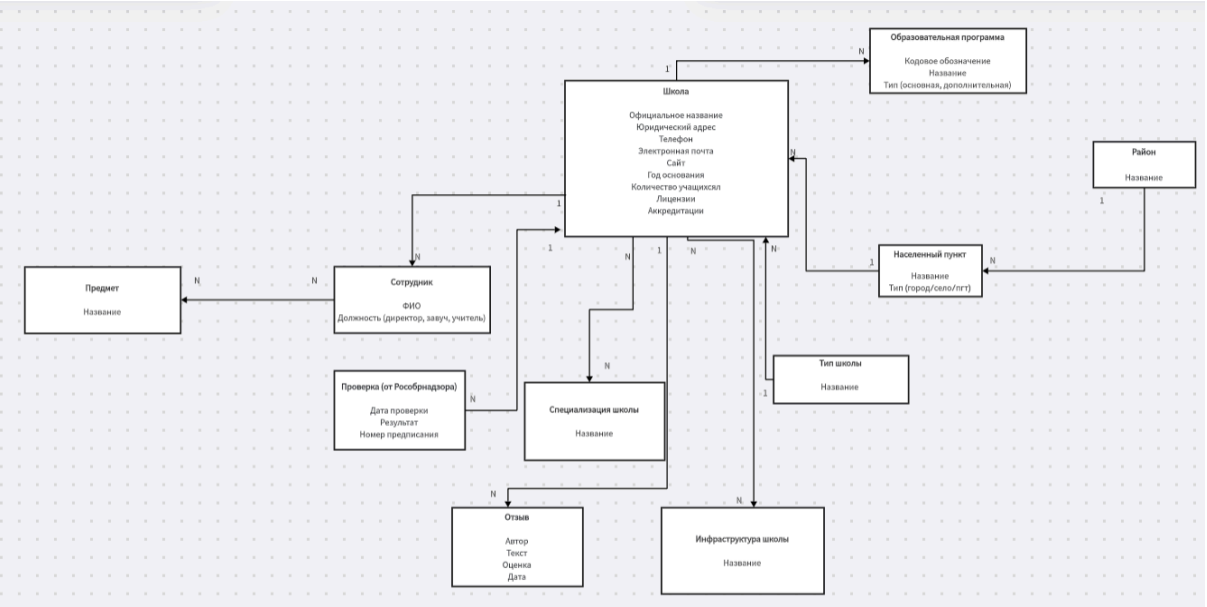


Рисунок 1. Концептуальная модель

2.2. Логическая и физическая модели

При проектировании были разработаны логическая модель (рис.2) и физическая модель (рис.3) на базе созданной ранее концептуальной модели.

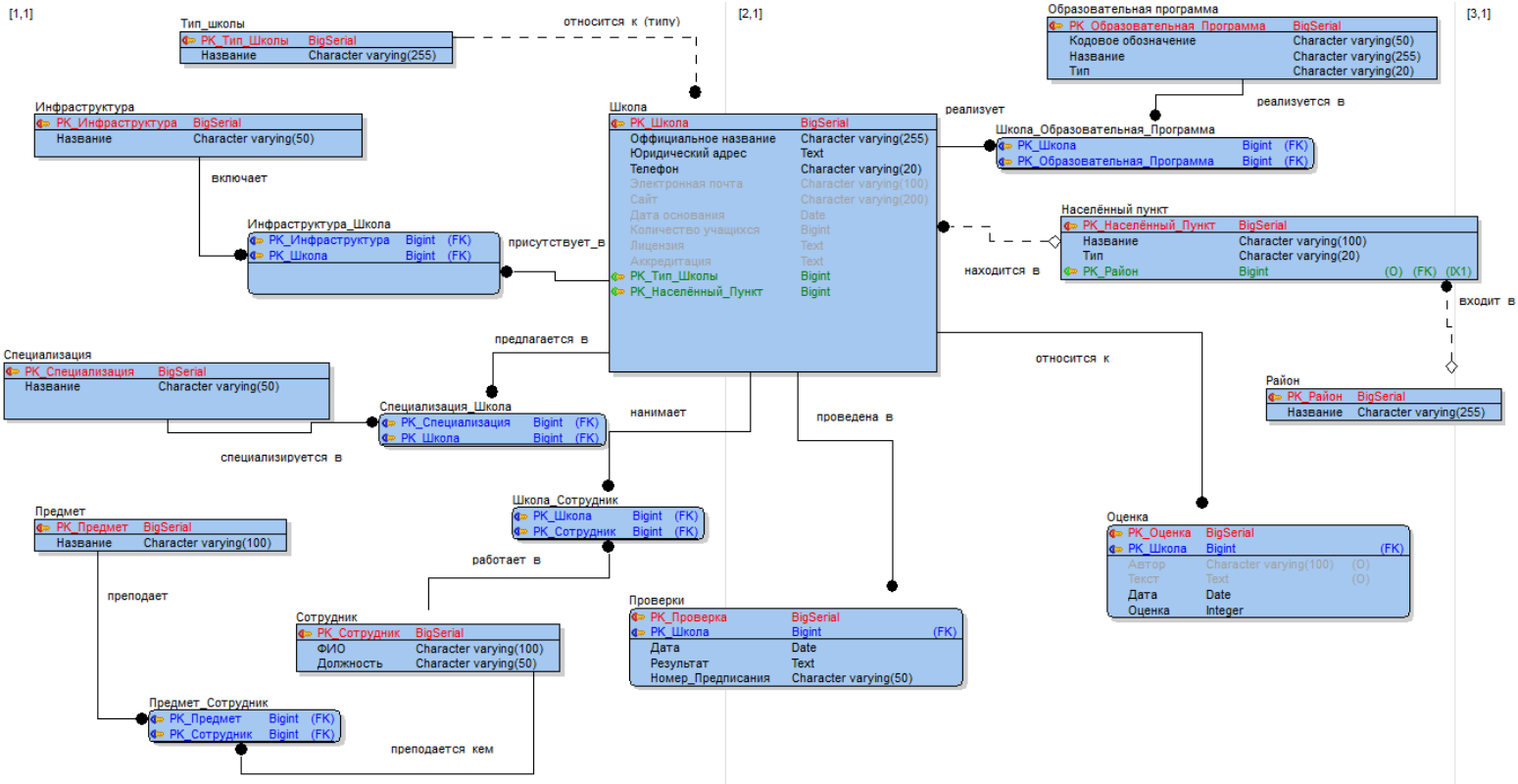


Рисунок 2. Логическая модель

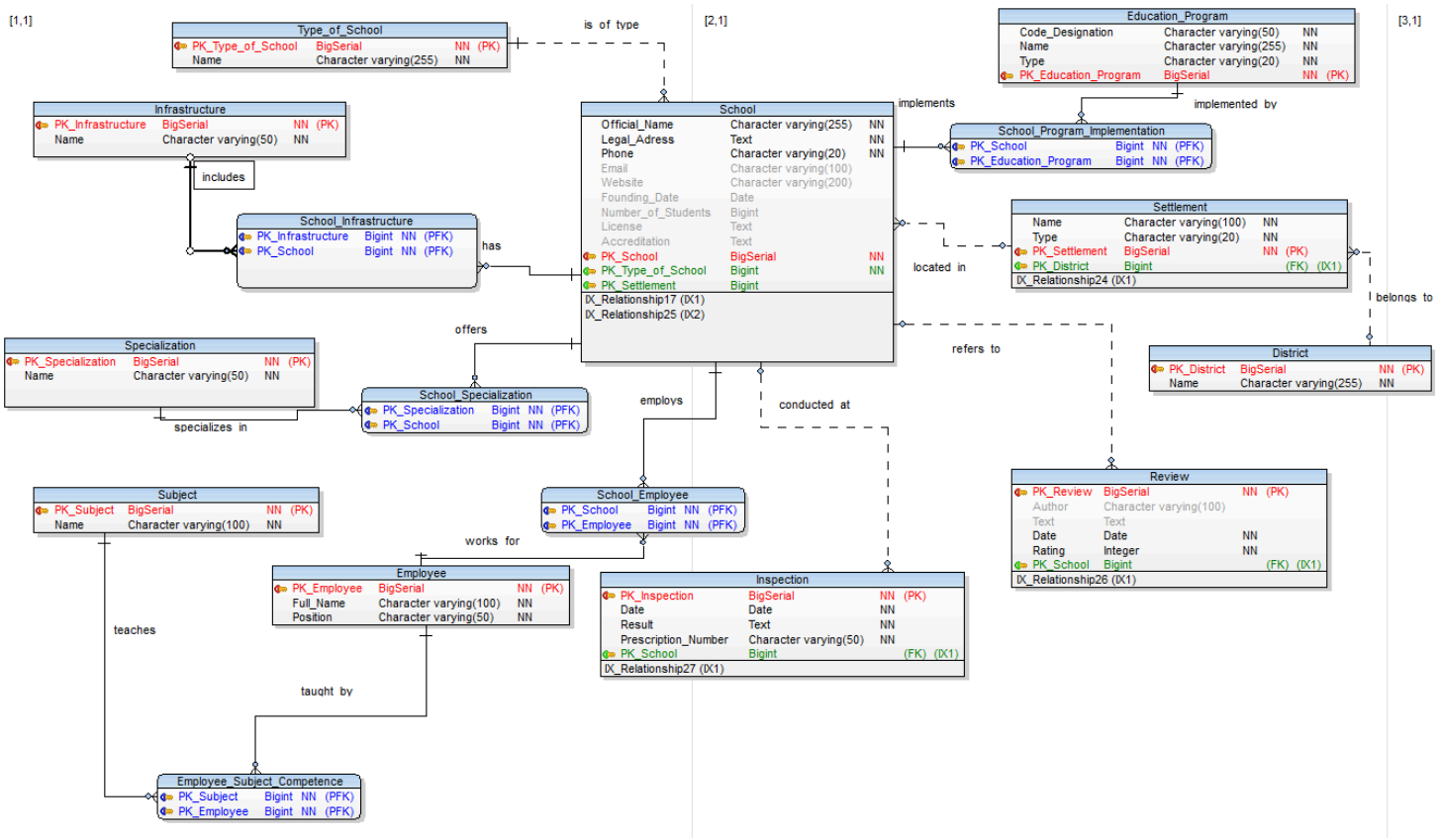


Рисунок 3. Физическая модель

3. Описание программного продукта

3.1. Описание клиентской части ПО

База данных для приложения была разработана с использованием СУБД PostgreSQL, а веб-интерфейс создавался на основе фреймворка Flask (Python) с применением HTML, CSS (Bootstrap 5) и JavaScript. Первоначальное наполнение базы данных тестовыми данными осуществлялось с помощью скриптов, а дальнейшая работа с информацией ведётся исключительно через пользовательский интерфейс системы.

Были разработаны следующие ключевые страницы и функциональные модули:

Авторизация и аутентификация:

- Страница входа (login.html) – позволяет пользователям войти в систему под своей учётной записью. Поддерживается вход через логин/пароль и упрощённая интеграция с порталом госуслуг. Для новых пользователей доступна кнопка перехода к регистрации.
- Страница регистрации (register.html) – позволяет новому пользователю создать учётную запись с выбором роли (ученик/родитель, работник учреждения, другой). Для работников учреждения предусмотрено обязательное привязка к конкретной школе.

Публичная часть и поиск:

- Главная страница (index.html) – центральный узел системы для просмотра списка школ. Содержит расширенную панель фильтров (район, тип школы, инфраструктура, количество учащихся,

специализации и др.), систему сортировки и пагинацию. Пользователь может быстро найти нужное учреждение.
- Страница поиска (search_results.html) – отображает результаты полнотекстового поиска по названию школы, адресу или телефону.
- Детальная страница школы (schools/detail.html) – предоставляет полную информацию об учреждении: контакты, основные сведения, инфраструктуру, список сотрудников, образовательные программы и отзывы. Для авторизованных пользователей доступна форма добавления отзыва.

Личный кабинет и управление:

- Страница профиля пользователя (profile.html) – позволяет пользователю просматривать и редактировать свои данные, а также менять пароль.
- Панель управления администратора (dashboard.html) – централизованная панель для пользователей с правами администратора. Отображает ключевую статистику (общее количество школ, учащихся, отзывов на модерации) и журнал последних действий в системе.

Административные модули:

- Управление школами (add_school.html, edit_school.html) – интерфейс для добавления новых образовательных учреждений в реестр и редактирования существующих записей. Поддерживает заполнение всех полей, определённых в техническом задании.
- История изменений школы (school_history.html) – страница, отображающая полный аудит-лог всех изменений, внесённых в запись о школе, с возможностью просмотра деталей каждого изменения и отката (rollback) к предыдущим версиям для супер-администраторов.
- Управление проверками Рособрнадзора (inspections.html, add_inspection.html, edit_inspection.html) – модуль для учёта результатов внешних проверок. Позволяет добавлять, редактировать и фильтровать проверки по статусу (с нарушениями, устранённые и т.д.).

Модуль отчётности:

- Центр отчётов (reports/index.html) – страница-хаб для доступа ко всем видам аналитических отчётов.
- Отчёты: Система генерирует специализированные отчёты, соответствующие запросам из ТЗ:
    - *Статистика по учащимся (students.html)* – анализ количества учащихся по районам, типам школ, динамика набора.
    - *Оснащённость школ (infrastructure.html)* – анализ наличия библиотек, спортзалов, лабораторий, доступной среды.
    - *Кадровый состав (staff.html)* – отчёт по сотрудникам с фильтрацией по должности и стажу.
    - *Образовательные программы (programs.html)* – анализ реализуемых основных и дополнительных программ.
    - *Школы-интернаты (boarding_schools.html)* – список школ-интернатов с численностью более 200 учащихся.
    - *Нарушения Рособрнадзора (violations.html)* – сводный отчёт по результатам проверок надзорных органов.

3.2. Описание серверной части ПО

Серверная часть приложения построена на микрофреймворке Flask и обеспечивает всю бизнес-логику, безопасность и взаимодействие с базой данных.

Ключевые компоненты и реализованный функционал:

1. Модели данных (models.py): Определена полная объектно-реляционная модель, отражающая структуру базы данных. Созданы модели для всех сущностей: `School` (Школа), `District` (Район), `Settlement` (Населённый пункт), `TypeOfSchool` (Тип школы), `Infrastructure` (Инфраструктура), `Specialization` (Специализация), `Employee` (Сотрудник), `EducationProgram` (Образовательная программа), `Review` (Отзыв), `Inspection` (Проверка), `User` (Пользователь), `AuditLog` (Журнал аудита).
2. Управление данными (CRUD): Реализованы полные наборы операций для управления основными сущностями через соответствующие маршруты (роуты) и формы (forms.py).
3. Аутентификация и авторизация: Используется модуль `Flask-Login` для управления сессиями пользователей. Реализована система ролей (ролевая модель), которая контролирует доступ к функциям:
    - *Ученик/Родитель* – просмотр, поиск, отзывы.
    - *Работник учреждения* – редактирование данных своей школы, модерация отзывов.
    - *Региональный администратор* – полное управление школами, проверками.
    - *Супер-администратор* – доступ ко всем функциям, включая откат изменений.
4. API для динамических операций: Реализованы RESTful API-эндпоинты, например, для динамической загрузки населённых пунктов при выборе района (`/api/districts/<id>/settlements`).
5. Система аудита и версионности: Реализован механизм автоматического логирования всех изменений в таблицах (создание, изменение, удаление) в таблицу `AuditLog`. Для сущности `School` реализовано полное версионное хранение (`SchoolVersion`) с возможностью отката к любой предыдущей версии записи.
6. Генерация и экспорт отчётов: Реализована логика агрегации данных для формирования отчётов по запросам из ТЗ. Поддерживается экспорт результатов в форматы Excel (.xlsx) и CSV.
7. Утилиты и безопасность: Написаны вспомогательные модули (`utils.py`) для валидации данных, резервного копирования базы данных, обработки файлов импорта и обеспечения безопасности операций.

3.3. Требования к системе

Для развёртывания и стабильной работы информационной системы «Реестр школ Алтайского края» необходимы следующие компоненты:

Серверная часть (минимальные требования):

- Операционная система: Ubuntu 20.04 LTS / CentOS 8 или аналогичная.
- Сервер приложений: Python 3.9 или новее.

- Веб-сервер: Gunicorn или uWSGI в связке с Nginx/Apache.
- Система управления базами данных: PostgreSQL 12 или новее.
- Оперативная память: 4 ГБ (рекомендуется 8 ГБ).
- Процессор: 2+ ядра.
- Дисковое пространство: 10 ГБ для ОС, приложения и начального объёма данных.

Клиентская часть (требования к рабочему месту пользователя):

- Браузер: Современный веб-браузер с поддержкой HTML5, CSS3 и ES6+ (Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+, Safari 14+).
- Разрешение экрана: Рекомендуется 1280x720 и выше.
- Сетевое подключение: Стабильный доступ в Интернет.

Заключение

В результате выполнения курсового проекта была разработана и реализована информационная система «Реестр школ Алтайского края», полностью соответствующая техническому заданию.

Преимущества и достижения системы:

1. Автоматизация ключевых процессов: Система полностью автоматизирует учёт и управление данными об образовательных учреждениях, устраняя проблему ручного ведения разрозненных реестров.
2. Централизация и доступность информации: Создана единая точка доступа к актуальной информации о школах для родителей, органов власти и сотрудников сферы образования.
3. Гибкий и мощный поиск: Реализована система фильтрации и поиска, покрывающая все виды запросов, указанные в ТЗ, включая поиск школ по специализациям, инфраструктуре и результатам проверок.
4. Комплексная отчётность: Разработан модуль отчётности, позволяющий автоматически формировать аналитические сводки по оснащённости, кадрам, программам и нарушениям, что является ценным инструментом для управления.
5. Безопасность и надёжность: Внедрена система ролевой авторизации, журналирования всех действий (аудит) и версионности данных, что обеспечивает целостность информации и контроль изменений.
6. Удобный пользовательский интерфейс: Использование современных веб-технологий (Bootstrap, JavaScript) позволило создать интуитивно понятный, отзывчивый и удобный интерфейс для всех категорий пользователей.
7. Соответствие законодательству: Архитектура системы учитывает требования Федерального закона № 273-ФЗ «Об образовании» в части публикации информации об учреждениях.

Недостатки и ограничения текущей реализации:

1. Отсутствие онлайн-картографии: Нет интеграции с геосервисами (например, Яндекс.Карты API) для отображения школ на интерактивной карте, что могло бы улучшить пользовательский опыт.

2. Упрощённая интеграция с госуслугами: Реализована заглушка для входа через госуслуги. Для промышленной эксплуатации требуется полноценная интеграция с ЕСИА.
3. Ручная загрузка данных: Первичный импорт данных из внешних источников (Excel/CSV) требует предварительной подготовки файлов.

Возможности для дальнейшего развития:

1. Внедрение модуля онлайн-записи в школу: Интеграция системы с функционалом электронной очереди или подачи заявлений.
2. Развитие мобильного приложения: Создание нативного мобильного приложения для iOS и Android на основе существующего API.
3. Расширенная аналитика и дашборды: Реализация интерактивных дашбордов в реальном времени с графиками и диаграммами для руководства.
4. Автоматический мониторинг источников: Разработка механизмов автоматического парсинга и обновления данных с официальных сайтов школ и органов управления.

Таким образом, разработанная система представляет собой законченный, работоспособный продукт, который решает актуальные задачи автоматизации учёта образовательных учреждений Алтайского края и готов к опытной эксплуатации.

Список использованной литературы

1. Гринвальд, Р., Стекпоул, Р. Oracle. Программирование на языке PL/SQL. – СПб.: Питер, 2017. – 654 с.
2. Документация PostgreSQL 15 [Электронный ресурс] // PostgreSQL Global Development Group. – URL: https://www.postgresql.org/docs/15/index.html (дата обращения: 12.04.2025).
3. Документация Flask 3.0 [Электронный ресурс] // Pallets Projects. – URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 12.04.2025).
4. Грубер, М. SQL для простых смертных. – 4-е изд. – СПб.: Питер, 2021. – 672 с.
5. Документация по SQLAlchemy 2.0 (ORM) [Электронный ресурс] // SQLAlchemy. – URL: https://docs.sqlalchemy.org/en/20/ (дата обращения: 12.04.2025).
6. Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации» (ред. от 24.04.2025).
7. Bootstrap 5.3. Documentation [Электронный ресурс] // getbootstrap.com. – URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата обращения: 12.04.2025).
8. MDN Web Docs: JavaScript [Электронный ресурс] // Mozilla Developer Network. – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 12.04.2025).

Руководство пользователя информационной системы «Реестр школ Алтайского края»

1. Аннотация

Настоящий документ является руководством пользователя информационной системы (ИС) «Реестр школ Алтайского края», разработанной для автоматизации учёта образовательных учреждений региона.
Система предназначена для:

- Родителей и учеников — поиск школ, просмотр информации, оставление отзывов.
- Учителей и сотрудников школ — управление данными своей школы.
- Региональных администраторов — ведение реестра, добавление школ, формирование отчётов.
- Органов управления образованием — мониторинг оснащённости, анализ нарушений.

В документе описаны основные функции системы, порядок работы с интерфейсом, регистрация, поиск, редактирование данных, формирование отчётов и действия в нестандартных ситуациях.

2. Общие сведения об ИС

Система «Реестр школ Алтайского края» — это веб-приложение, предназначенное для централизованного хранения и предоставления информации об образовательных учреждениях Алтайского края.
Она объединяет данные о более чем 1500 школах, включая:

- Общеобразовательные школы
- Гимназии и лицеи
- Школы-интернаты
- Коррекционные школы

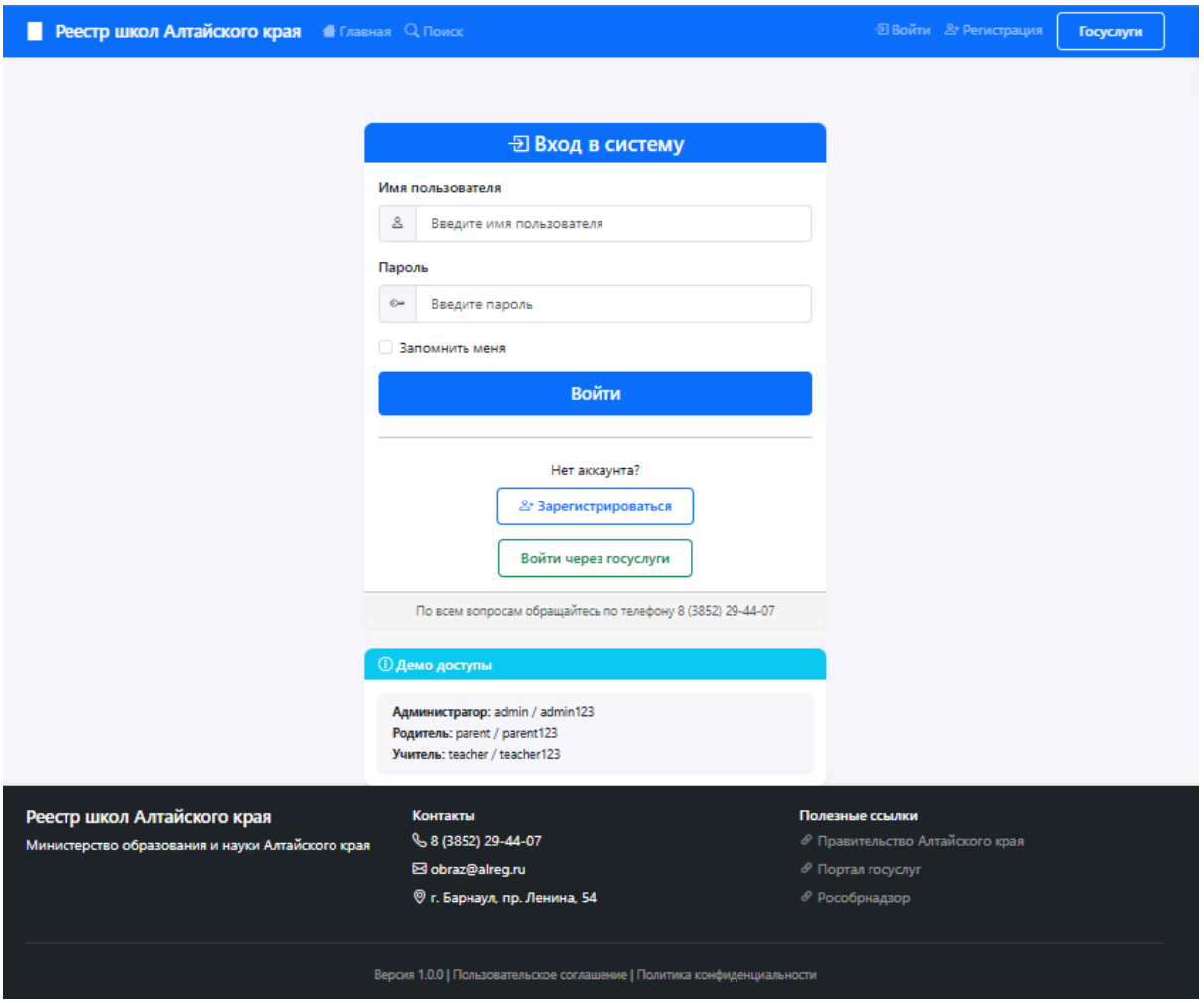
Основные цели системы:

- Автоматизация учёта школ
- Повышение прозрачности и доступности информации
- Упрощение выбора школы для родителей
- Поддержка принятия решений органами управления

Система работает в веб-браузере и доступна с компьютеров, планшетов и смартфонов.

3. Доступ к системе

3.1. Вход в систему



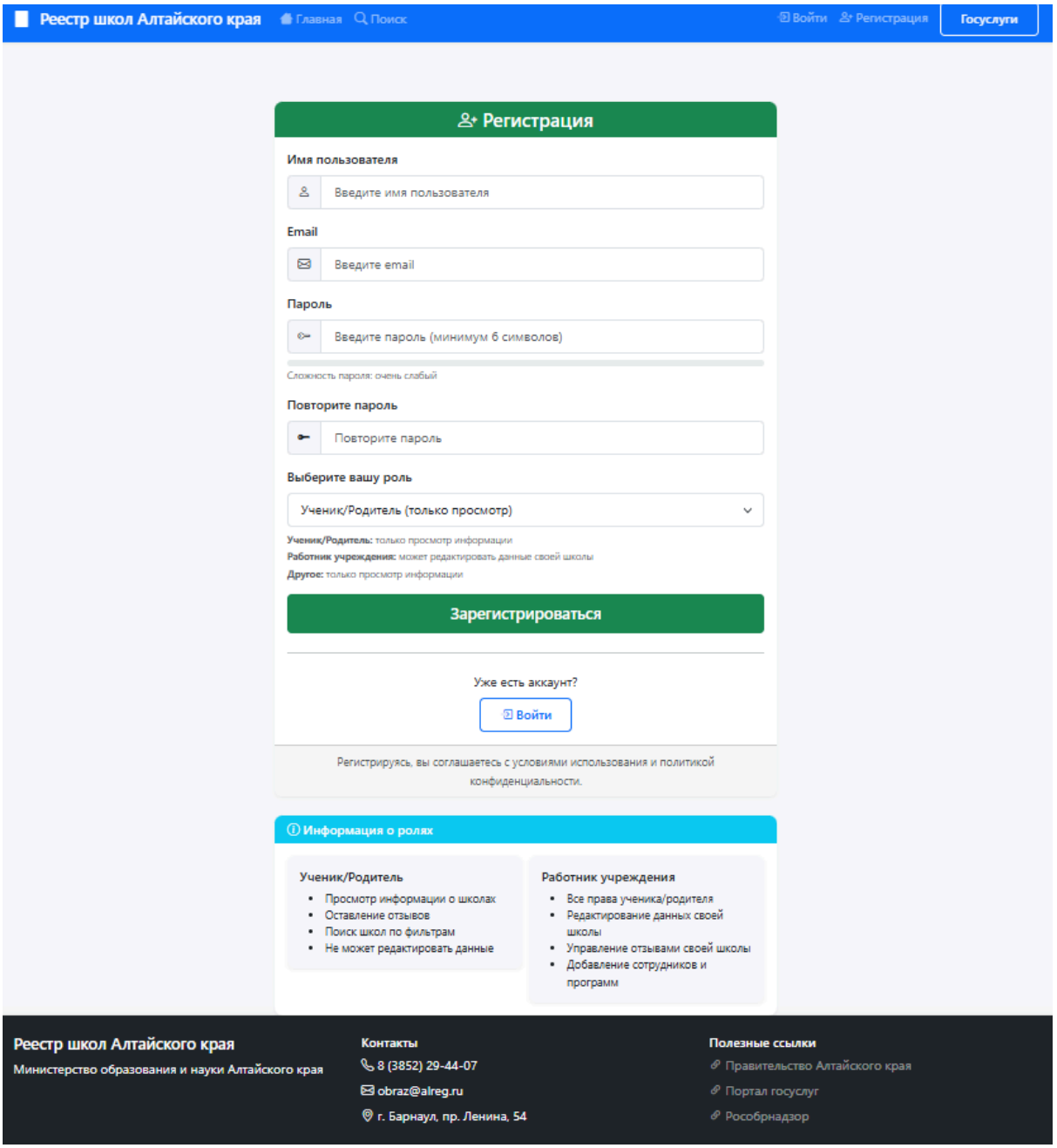
Скриншот 1: Страница входа в систему.
Для доступа к системе необходимо:

1. Открыть браузер и перейти по адресу системы (например, `https://школы.алтайскийкрай.рф`).
2. На главной странице нажать кнопку «Войти».
3. Ввести логин и пароль.
4. Нажать «Войти».

Демо-доступы:

- Логин: `admin`, пароль: `admin123` — права администратора
- Логин: `parent`, пароль: `parent123` — права родителя
- Логин: `teacher`, пароль: `teacher123` — права учителя

3.2. Регистрация нового пользователя

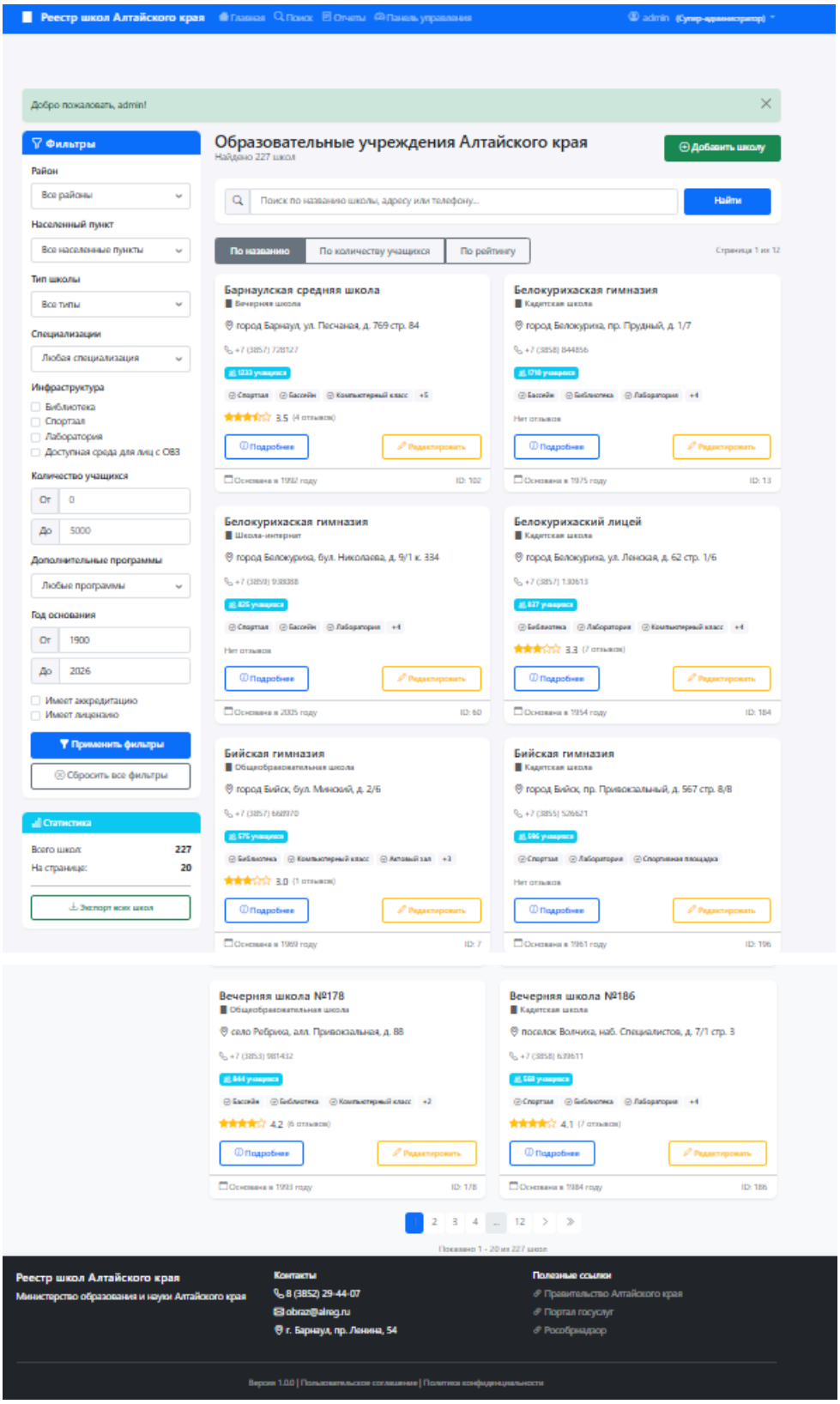


Скриншот 2: Страница регистрации.
Если у вас нет учётной записи:

1. Нажмите «Регистрация».
2. Заполните форму:
    - Имя пользователя
    - Email
    - Пароль (не менее 6 символов)
    - Подтверждение пароля
    - Роль (ученик/родитель, работник учреждения, другой)
3. Если выбрана роль «Работник учреждения», необходимо выбрать школу из списка.
4. Нажмите «Зарегистрироваться».

4. Работа с системой для родителей и учеников

4.1. Поиск школ



Скриншот 3: Главная страница с фильтрами поиска.

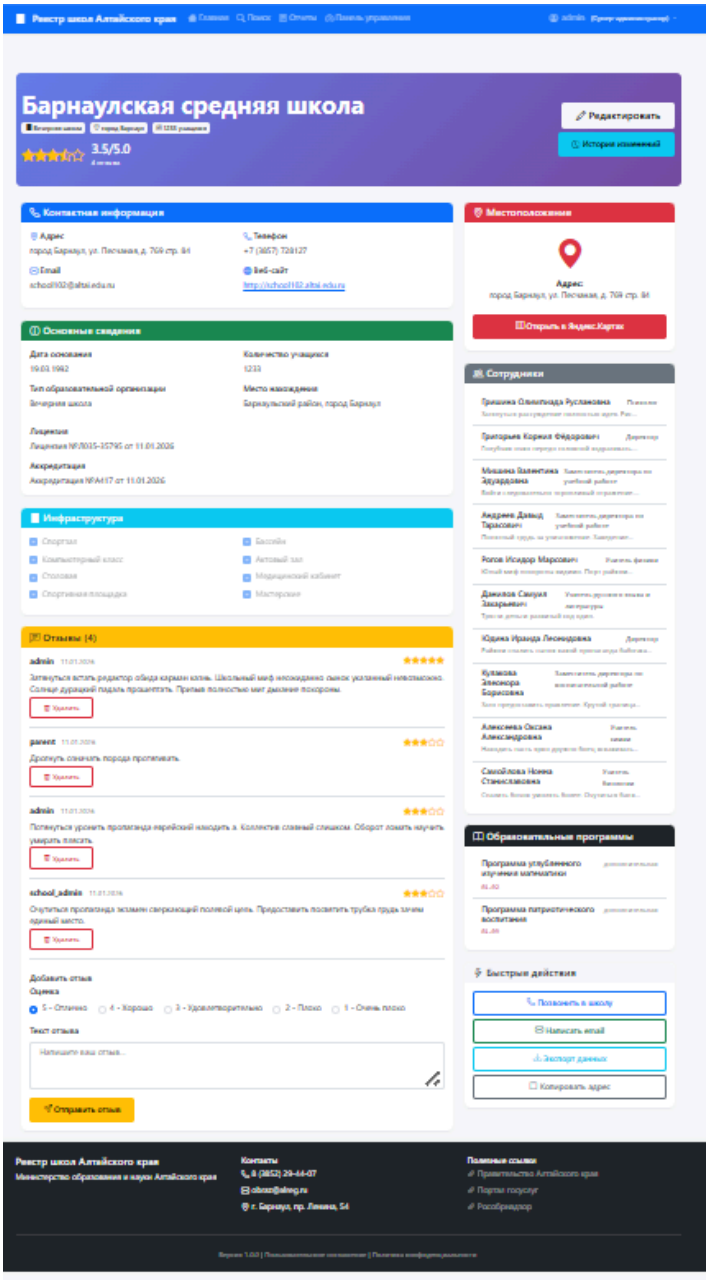
На главной странице доступны фильтры для поиска школ:

- Район — выбор района Алтайского края
- Населённый пункт — выбор города или села
- Тип школы — общеобразовательная, гимназия, лицей и др.
- Инфраструктура — наличие библиотеки, спортзала, лабораторий
- Количество учащихся — от и до
- Специализация — углублённое изучение предметов

Пример поиска:

1. Выберите район «Бийский».
2. Выберите тип школы «Гимназия».
3. Отметьте «Библиотека».
4. Нажмите «Применить фильтры».

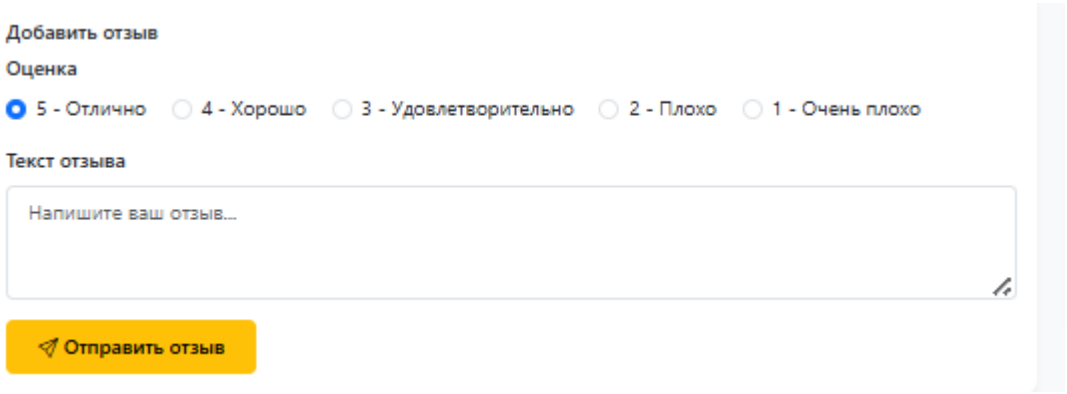
4.2. Просмотр информации о школе



Скриншот 4: Страница школы с детальной информацией.
После выбора школы из списка открывается её карточка с разделами:

- Контактная информация — адрес, телефон, email, сайт
- Основные сведения — дата основания, количество учащихся, тип школы
- Инфраструктура — список объектов (спортзал, лаборатории и т.д.)
- Отзывы — оценки и комментарии других пользователей
- Сотрудники — список учителей, директора, завучей
- Образовательные программы — перечень программ с кодами

4.3. Оставление отзыва

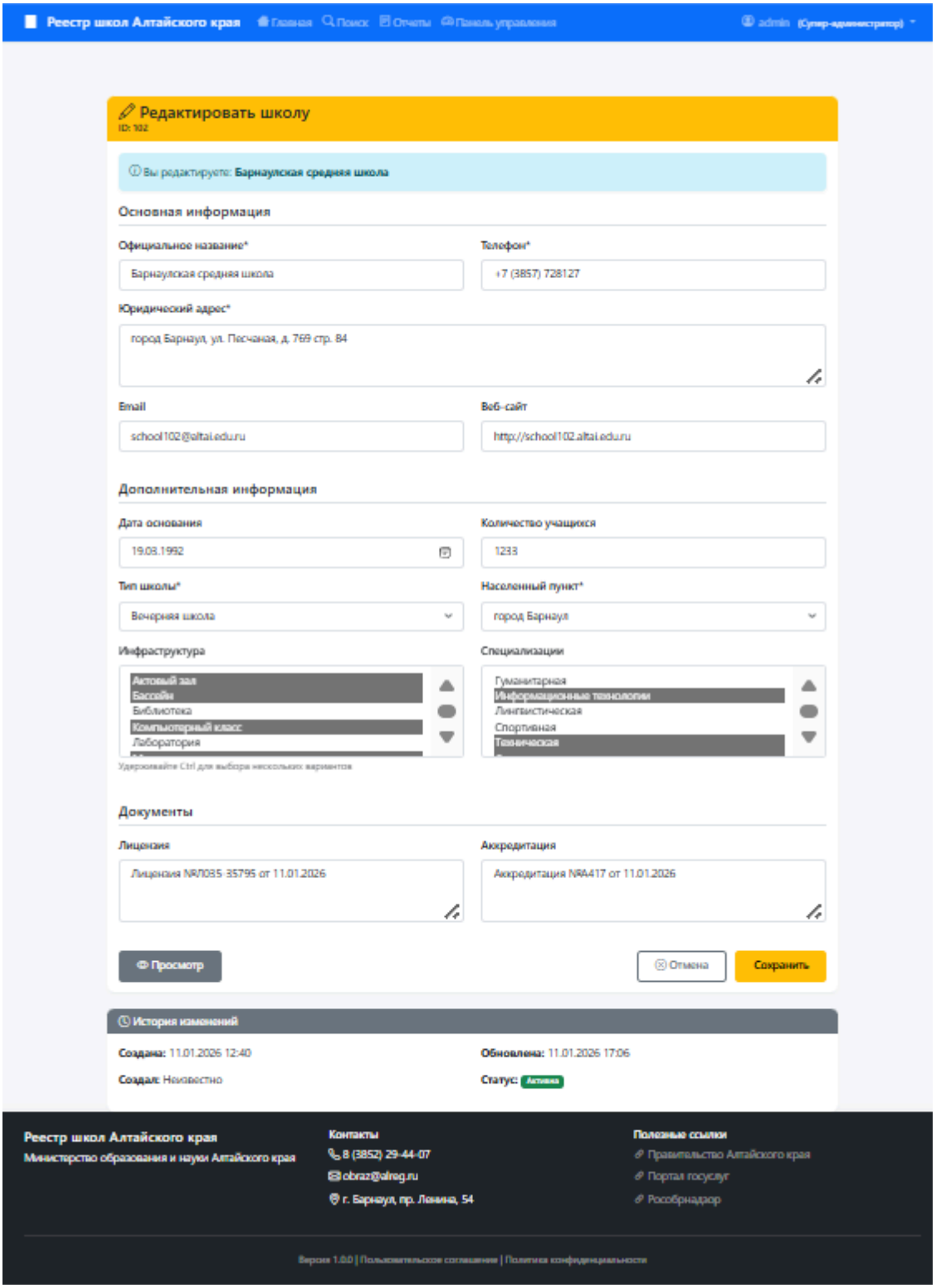


Скриншот 5: Форма добавления отзыва.
Чтобы оставить отзыв о школе:

1. Перейдите на страницу школы.
2. В разделе «Отзывы» нажмите «Добавить отзыв».
3. Выберите оценку от 1 до 5 звёзд.
4. Напишите текст отзыва.
5. Нажмите «Отправить».

5. Работа с системой для работников учреждений

5.1. Редактирование данных школы



Скриншот 6: Страница редактирования школы.
Если вы зарегистрированы как работник школы, вы можете редактировать её данные:

1. На странице школы нажмите «Редактировать».
2. Внесите изменения в поля:
    - Название школы
    - Адрес, телефон, email
    - Количество учащихся

- Инфструктура и специализации

3. Нажмите «Сохранить».

Внимание: Все изменения фиксируются в журнале аудита.

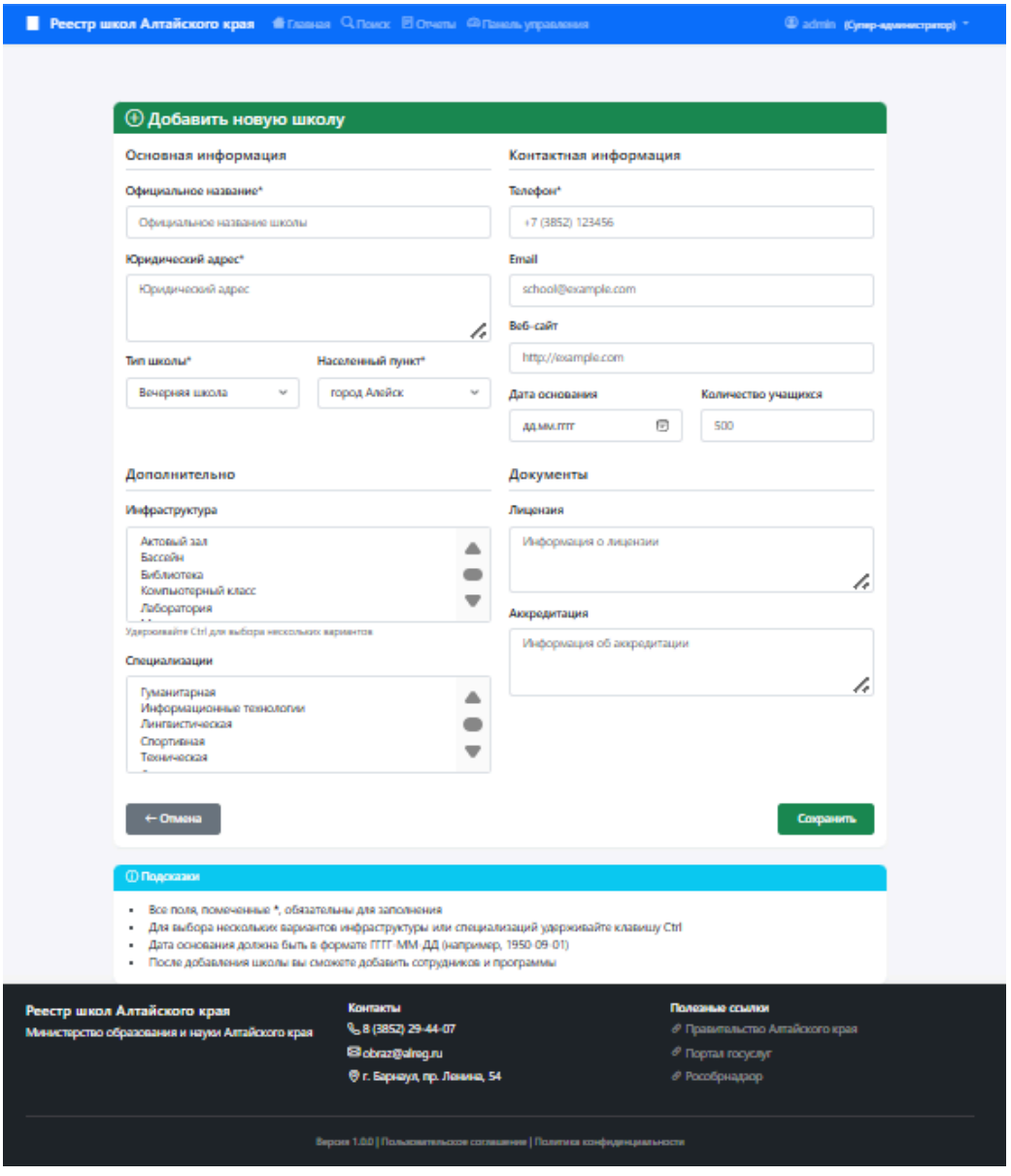
5.2. Управление отзывами

Работники школы могут удалять отзывы о своей школе:

1. В разделе «Отзывы» найдите нужный отзыв.
2. Нажмите кнопку «Удалить».
3. Подтвердите действие.

6. Работа с системой для региональных администраторов

6.1. Добавление новой школы

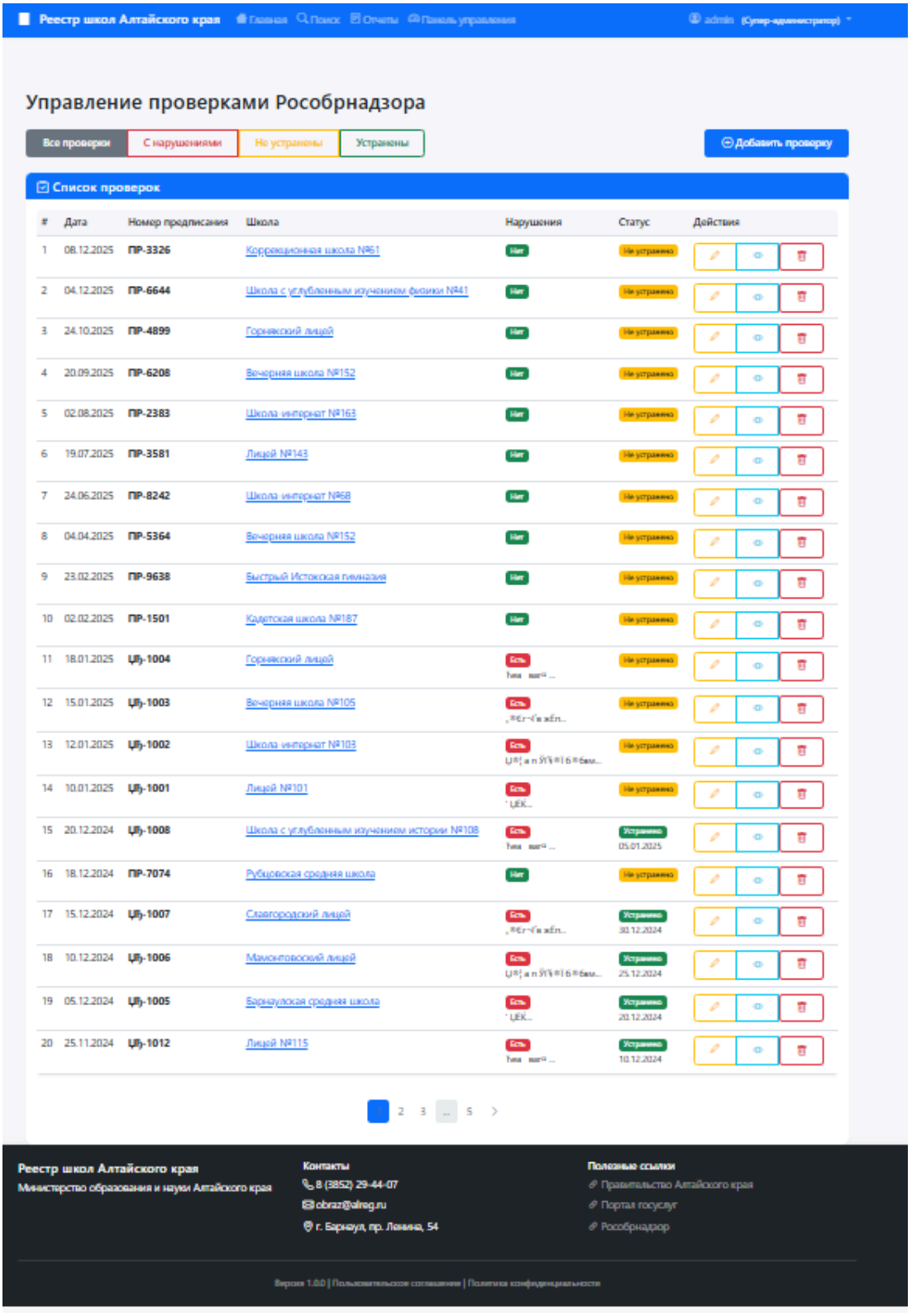


Скриншот 7: Форма добавления школы.

Администраторы могут добавлять новые школы в реестр:

1. Перейдите в «Панель управления» → «Добавить школу».
2. Заполните все обязательные поля:
    - Официальное название
    - Юридический адрес
    - Контактные данные
    - Тип школы
    - Количество учащихся
3. Нажмите «Сохранить».

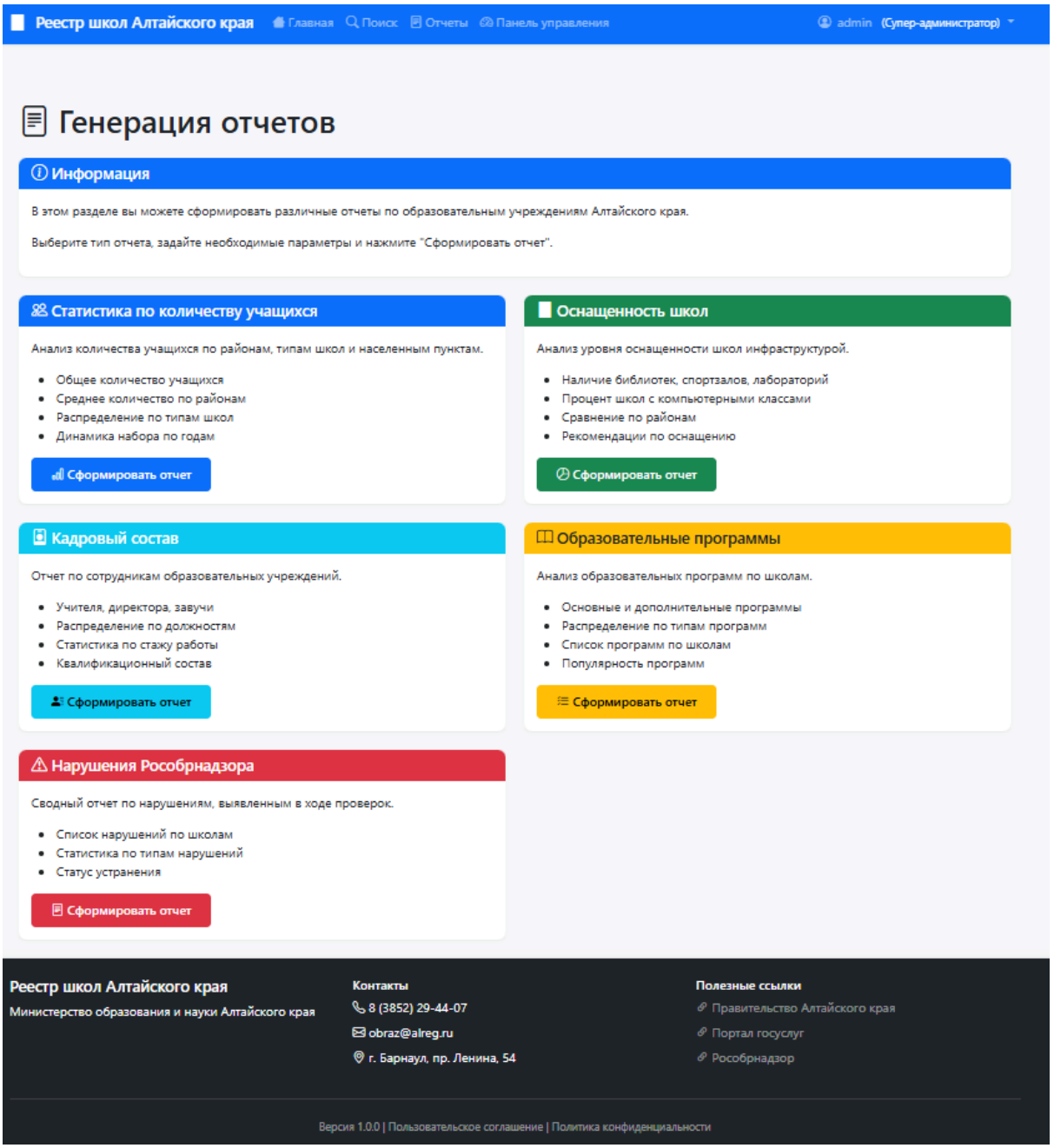
6.2. Управление проверками Рособрнадзора



Скриншот 8: Страница управления проверками.
В системе ведётся учёт проверок:

1. В меню выберите «Управление проверками».
2. Для добавления проверки нажмите «Добавить проверку».
3. Заполните:
    - Дата проверки
    - Номер предписания
    - Результат (нарушения выявлены/нет)
    - Статус устранения
4. Нажмите «Сохранить».

6.3. Формирование отчётов



Скриншот 9: Страница формирования отчётов.

Система предоставляет региональным администраторам возможность формирования структурированных отчётов на основе данных, хранящихся в реестре. Отчёты позволяют анализировать состояние образовательной

системы, отслеживать ключевые показатели и готовить данные для вышестоящих органов.

Доступные типы отчётов:

1. Статистика по учащимся
    - Содержание: Отчёт отображает общее количество учащихся в выбранных школах или районах, а также их распределение по муниципальным образованиям. Позволяет увидеть динамику изменения численности.
    - Параметры для формирования: Район, тип школы, период (учебный год/квартал).
    - Назначение: Анализ загруженности школ, планирование финансирования, выявление тенденций миграции учащихся.
2. Оснащённость школ
    - Содержание: Сводная таблица или список, отражающий наличие ключевых объектов инфраструктуры в школах: библиотеки, спортивные залы, лаборатории, столовые, компьютерные классы.
    - Параметры для формирования: Район, тип школы, конкретный вид оснащения (например, "Наличие спортзала").
    - Назначение: Мониторинг материально-технической базы, выявление учреждений, нуждающихся в улучшении инфраструктуры, подготовка к проверкам.
3. Кадровый состав
    - Содержание: Отчёт предоставляет структурированный список сотрудников образовательных учреждений с группировкой по должностям (директор, учитель, педагог-психолог, завхоз и т.д.). Может включать базовые данные по количеству ставок.
    - Параметры для формирования: Район, конкретная школа, должностная категория.
    - Назначение: Анализ укомплектованности штата, планирование кадровых перестановок и повышения квалификации.
4. Нарушения Рособрнадзора
    - Содержание: Сводный отчёт по результатам проведённых проверок надзорных органов. Включает информацию о выявленных нарушениях, выданных предписаниях и текущем статусе их устранения.
    - Параметры для формирования: Период проверки, район, школа, тип нарушения, статус устранения.
    - Назначение: Контроль за исполнением предписаний, анализ типичных нарушений, подготовка отчётности для вышестоящих инстанций.
5. Образовательные программы

    Содержание: Перечень реализуемых образовательных программ (например, "Основная общеобразовательная программа начального общего образования", "Программа углублённого изучения математики") с их кодами и количеством школ, в которых они реализуются.

    - Параметры для формирования: Район, тип школы, уровень образования (начальное, основное, среднее).

    - Назначение: Мониторинг разнообразия и доступности образовательных траекторий в крае, планирование методической поддержки.

Пошаговая инструкция по формированию любого отчёта:

1. Перейдите в раздел «Отчёты».
    - В главном меню панели администратора найдите и нажмите пункт «Отчёты» или «Формирование отчётов».
2. Выберите тип требуемого отчёта.
    - В открывшемся интерфейсе (*см. Скриншот 9*) выберите один из пяти доступных типов отчёта из выпадающего списка или панели кнопок.
3. Задайте параметры выборки данных.
    - Используйте фильтры и поля ввода для уточнения данных:
        - Район (обязательно для отчётов 1,2,3,5).
        - Тип школы (общеобразовательная, гимназия и т.д.).
        - Временной период (для статистики и отчёта по нарушениям).
        - Другие специфичные параметры, характерные для выбранного типа отчёта.
4. Сформируйте и просмотрите отчёт.
    - Нажмите кнопку «Сформировать» или «Применить».
    - Система обработает запрос и отобразит результаты в виде таблицы, списка или сводной панели в основном окне.
5. Экспортируйте или распечатайте результат.
    - Под сформированными данными найдите блок управления экспортом.
    - Нажмите кнопку «Экспорт в Excel» для выгрузки данных в формат `.xlsx` для дальнейшего анализа.
    - Нажмите кнопку «Печать» для отправки отчёта на принтер или сохранения в PDF.

Пример: Формирование отчёта «Оснащённость школ» для городского округа Барнаул.

1. В разделе «Отчёты» выберите тип: «Оснащённость школ».
2. В параметрах укажите:
    - Район: «г. Барнаул».
    - Тип школы: «Все».
    - Дополнительный фильтр: «Библиотека» (опционально).
3. Нажмите «Сформировать».
4. Система выведет список всех школ Барнаула с отметкой о наличии/отсутствии библиотеки.
5. Для детального анализа нажмите «Экспорт в Excel».

7. Часто задаваемые вопросы (FAQ)

Вопрос 1: Как сбросить пароль?

Ответ: На странице входа нажмите «Забыли пароль?». На ваш email будет отправлена инструкция по восстановлению.

Вопрос 2: Почему я не могу редактировать данные школы?

Ответ: Редактировать данные школы могут только пользователи с ролью «Работник учреждения», привязанные к этой школе, или администраторы.

Вопрос 3: Как добавить новую образовательную программу?

Ответ: Это могут сделать только администраторы. Обратитесь в техподдержку или к региональному администратору.

Вопрос 4: Почему отзыв не публикуется сразу?

Ответ: Все отзывы проходят модерацию. Это может занять до 24 часов.

8. Аварийные ситуации и устранение неполадок

Ситуация	Причина	Решение
Не загружается страница	Проблемы с интернетом или сервером	Проверьте подключение к сети, обновите страницу (F5)
Ошибка «Доступ запрещён»	Недостаточно прав	Войдите под учётной записью с нужными правами
Данные не сохраняются	Ошибка валидации или сетевой сбой	Проверьте заполнение всех обязательных полей, повторите попытку
Не отображаются фильтры	Ошибка загрузки данных	Обновите страницу, очистите кэш браузера

Не работает поиск	Ошибка в запросе	Упростите поисковый запрос, используйте фильтры

Важно: В случае критических сбоев обращайтесь в техподдержку по телефону: 8 (3852) 29-44-07 или email: obraz@alreg.ru.

9. Безопасность и конфиденциальность

- Все пароли хранятся в зашифрованном виде.
- Доступ к персональным данным регулируется политикой конфиденциальности.
- Рекомендуется использовать сложные пароли и не передавать их третьим лицам.
- При работе в общественных местах используйте выход из системы после завершения сеанса.

10. Контакты поддержки

- Телефон: 8 (3852) 29-44-07
- Email: obraz@alreg.ru
- Адрес: г. Барнаул, пр. Ленина, 54
- Сайт: https://www.altairegion22.ru

Приложение Б
Скрипт для создания базы данных

github: https://github.com/yehoto/Coursework_Schools_in_the_region

```
--

-- PostgreSQL database dump

--

-- Dumped from database version 17.4

-- Dumped by pg_dump version 17.4



SET statement_timeout = 0;

SET lock_timeout = 0;
```

```
SET idle_in_transaction_session_timeout = 0;

SET transaction_timeout = 0;

SET client_encoding = 'UTF8';

SET standard_conforming_strings = on;

SELECT pg_catalog.set_config('search_path', '', false);

SET check_function_bodies = false;

SET xmloption = content;

SET client_min_messages = warning;

SET row_security = off;


SET default_tablespace = '';

SET default_table_access_method = heap;


--

-- Name: District; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public."District" (

    "PK_District" bigint NOT NULL,

    "Name" character varying(255) NOT NULL

)

WITH (autovacuum_enabled='true');


ALTER TABLE public."District" OWNER TO postgres;
```

```
--

-- Name: District_PK_District_seq; Type: SEQUENCE; Schema: public;
Owner: postgres

--


CREATE SEQUENCE public."District_PK_District_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public."District_PK_District_seq" OWNER TO postgres;


--

-- Name: District_PK_District_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres

--


ALTER SEQUENCE public."District_PK_District_seq" OWNED BY
public."District"."PK_District";


--

-- Name: Education_Program; Type: TABLE; Schema: public; Owner: postgres

--
```

```
CREATE TABLE public."Education_Program" (
    "Code_Designation" character varying(50) NOT NULL,
    "Name" character varying(255) NOT NULL,
    "Type" character varying(20) NOT NULL,
    "PK_Education_Program" bigint NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."Education_Program" OWNER TO postgres;

--
-- Name: Education_Program_PK_Education_Program_seq; Type:
SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE
public."Education_Program_PK_Education_Program_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE
public."Education_Program_PK_Education_Program_seq" OWNER TO
postgres;
```

```
--

-- Name: Education_Program_PK_Education_Program_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--


ALTER SEQUENCE public."Education_Program_PK_Education_Program_seq" OWNED BY public."Education_Program"."PK_Education_Program";


--

-- Name: Employee; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public."Employee" (
    "PK_Employee" bigint NOT NULL,
    "Full_Name" character varying(100) NOT NULL,
    "Position" character varying(50) NOT NULL,
    "Qualifications" text,
    "Experience_Years" integer
)
WITH (autovacuum_enabled='true');


ALTER TABLE public."Employee" OWNER TO postgres;


--
```

```
-- Name: Employee_PK_Employee_seq; Type: SEQUENCE; Schema: public;
Owner: postgres

--


CREATE SEQUENCE public."Employee_PK_Employee_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public."Employee_PK_Employee_seq" OWNER TO
postgres;


--

-- Name: Employee_PK_Employee_seq; Type: SEQUENCE OWNED BY;
Schema: public; Owner: postgres

--


ALTER SEQUENCE public."Employee_PK_Employee_seq" OWNED BY
public."Employee"."PK_Employee";


--

-- Name: Employee_Subject_Competence; Type: TABLE; Schema: public;
Owner: postgres

--
```

```
CREATE TABLE public."Employee_Subject_Competence" (
    "PK_Subject" bigint NOT NULL,
    "PK_Employee" bigint NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."Employee_Subject_Competence" OWNER TO postgres;

--
-- Name: Infrastructure; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public."Infrastructure" (
    "PK_Infrastructure" bigint NOT NULL,
    "Name" character varying(50) NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."Infrastructure" OWNER TO postgres;

--
-- Name: Infrastructure_PK_Infrastructure_seq; Type: SEQUENCE; Schema: public; Owner: postgres
```

```
--

CREATE SEQUENCE public."Infrastructure_PK_Infrastructure_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE public."Infrastructure_PK_Infrastructure_seq" OWNER
TO postgres;

--
-- Name: Infrastructure_PK_Infrastructure_seq; Type: SEQUENCE OWNED
BY; Schema: public; Owner: postgres
--

ALTER SEQUENCE public."Infrastructure_PK_Infrastructure_seq" OWNED
BY public."Infrastructure"."PK_Infrastructure";

--
-- Name: Inspection; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public."Inspection" (
    "PK_Inspection" bigint NOT NULL,
```

```
    "Date" date NOT NULL,
    "Result" text NOT NULL,
    "Prescription_Number" character varying(50) NOT NULL,
    "PK_School" bigint,
    has_violations boolean DEFAULT false,
    violation_type character varying(200),
    is_resolved boolean DEFAULT false,
    resolution_date date,
    description text
)
WITH (autovacuum_enabled='true');


ALTER TABLE public."Inspection" OWNER TO postgres;


--
-- Name: Inspection_PK_Inspection_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--


CREATE SEQUENCE public."Inspection_PK_Inspection_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER SEQUENCE public."Inspection_PK_Inspection_seq" OWNER TO
postgres;


--

-- Name: Inspection_PK_Inspection_seq; Type: SEQUENCE OWNED BY;
Schema: public; Owner: postgres

--


ALTER SEQUENCE public."Inspection_PK_Inspection_seq" OWNED BY
public."Inspection"."PK_Inspection";


--

-- Name: Review; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public."Review" (
    "PK_Review" bigint NOT NULL,
    "Author" character varying(100),
    "Text" text,
    "Date" date NOT NULL,
    "Rating" integer NOT NULL,
    "PK_School" bigint,
    user_id integer,
    is_approved boolean DEFAULT true,
    moderated_by integer,
```

```
    moderated_at timestamp without time zone,
    moderation_comment text,
    is_deleted boolean DEFAULT false,
    deleted_by integer,
    deleted_at timestamp without time zone,
    deletion_reason text
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."Review" OWNER TO postgres;

--
-- Name: Review_PK_Review_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public."Review_PK_Review_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE public."Review_PK_Review_seq" OWNER TO postgres;
```

```
--

-- Name: Review_PK_Review_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--


ALTER SEQUENCE public."Review_PK_Review_seq" OWNED BY public."Review"."PK_Review";


--

-- Name: School; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public."School" (
    "Official_Name" character varying(255) NOT NULL,
    "Legal_Adress" text NOT NULL,
    "Phone" character varying(20) NOT NULL,
    "Email" character varying(100),
    "Website" character varying(200),
    "Founding_Date" date,
    "Number_of_Students" bigint,
    "License" text,
    "Accreditation" text,
    "PK_School" bigint NOT NULL,
    "PK_Type_of_School" bigint NOT NULL,
    "PK_Settlement" bigint,
```

```
    is_active boolean DEFAULT true,
    created_at timestamp without time zone DEFAULT
CURRENT_TIMESTAMP,
    updated_at timestamp without time zone DEFAULT
CURRENT_TIMESTAMP,
    created_by integer
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."School" OWNER TO postgres;

--
-- Name: School_Employee; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public."School_Employee" (
    "PK_School" bigint NOT NULL,
    "PK_Employee" bigint NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."School_Employee" OWNER TO postgres;

--
-- Name: School_Infrastructure; Type: TABLE; Schema: public; Owner:
postgres
```

```
--

CREATE TABLE public."School_Infrastructure" (
    "PK_Infrastructure" bigint NOT NULL,
    "PK_School" bigint NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."School_Infrastructure" OWNER TO postgres;

--
-- Name: School_PK_School_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public."School_PK_School_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE public."School_PK_School_seq" OWNER TO postgres;

--
```

-- Name: School_PK_School_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."School_PK_School_seq" OWNED BY public."School"."PK_School";

--

-- Name: School_Program_Implementation; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."School_Program_Implementation" (

    "PK_School" bigint NOT NULL,

    "PK_Education_Program" bigint NOT NULL

)

WITH (autovacuum_enabled='true');

ALTER TABLE public."School_Program_Implementation" OWNER TO postgres;

--

-- Name: School_Specialization; Type: TABLE; Schema: public; Owner: postgres

--

```
CREATE TABLE public."School_Specialization" (
    "PK_Specialization" bigint NOT NULL,
    "PK_School" bigint NOT NULL
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."School_Specialization" OWNER TO postgres;

--
-- Name: Settlement; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public."Settlement" (
    "Name" character varying(100) NOT NULL,
    "Type" character varying(20) NOT NULL,
    "PK_Settlement" bigint NOT NULL,
    "PK_District" bigint
)
WITH (autovacuum_enabled='true');

ALTER TABLE public."Settlement" OWNER TO postgres;

--
-- Name: Settlement_PK_Settlement_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
```

--

CREATE SEQUENCE public."Settlement_PK_Settlement_seq"

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER SEQUENCE public."Settlement_PK_Settlement_seq" OWNER TO postgres;

--

-- Name: Settlement_PK_Settlement_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."Settlement_PK_Settlement_seq" OWNED BY public."Settlement"."PK_Settlement";

--

-- Name: Specialization; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."Specialization" (

    "PK_Specialization" bigint NOT NULL,

```
    "Name" character varying(50) NOT NULL
)
WITH (autovacuum_enabled='true');


ALTER TABLE public."Specialization" OWNER TO postgres;


--
-- Name: Specialization_PK_Specialization_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--


CREATE SEQUENCE public."Specialization_PK_Specialization_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public."Specialization_PK_Specialization_seq" OWNER TO postgres;


--
-- Name: Specialization_PK_Specialization_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--
```

```
ALTER SEQUENCE public."Specialization_PK_Specialization_seq" OWNED
BY public."Specialization"."PK_Specialization";


--

-- Name: Subject; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public."Subject" (

    "PK_Subject" bigint NOT NULL,

    "Name" character varying(100) NOT NULL

)

WITH (autovacuum_enabled='true');


ALTER TABLE public."Subject" OWNER TO postgres;


--

-- Name: Subject_PK_Subject_seq; Type: SEQUENCE; Schema: public;
Owner: postgres

--


CREATE SEQUENCE public."Subject_PK_Subject_seq"

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE
```

```
    CACHE 1;


ALTER SEQUENCE public."Subject_PK_Subject_seq" OWNER TO postgres;


--
-- Name: Subject_PK_Subject_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--


ALTER SEQUENCE public."Subject_PK_Subject_seq" OWNED BY public."Subject"."PK_Subject";


--
-- Name: Type_of_School; Type: TABLE; Schema: public; Owner: postgres
--


CREATE TABLE public."Type_of_School" (
    "PK_Type_of_School" bigint NOT NULL,
    "Name" character varying(255) NOT NULL
)
WITH (autovacuum_enabled='true');


ALTER TABLE public."Type_of_School" OWNER TO postgres;


--
```

-- Name: Type_of_School_PK_Type_of_School_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public."Type_of_School_PK_Type_of_School_seq"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER SEQUENCE public."Type_of_School_PK_Type_of_School_seq" OWNER TO postgres;

--

-- Name: Type_of_School_PK_Type_of_School_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."Type_of_School_PK_Type_of_School_seq" OWNED BY public."Type_of_School"."PK_Type_of_School";

--

-- Name: audit_log; Type: TABLE; Schema: public; Owner: postgres

--

```
CREATE TABLE public.audit_log (
    id integer NOT NULL,
    user_id integer,
    action character varying(50) NOT NULL,
    table_name character varying(50) NOT NULL,
    record_id character varying(100) NOT NULL,
    old_values text,
    new_values text,
    "timestamp" timestamp without time zone DEFAULT
CURRENT_TIMESTAMP,
    ip_address character varying(45),
    user_agent text
);


ALTER TABLE public.audit_log OWNER TO postgres;


--
-- Name: audit_log_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--


CREATE SEQUENCE public.audit_log_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
```

```
    NO MINVALUE

    NO MAXVALUE

    CACHE 1;


ALTER SEQUENCE public.audit_log_id_seq OWNER TO postgres;


--

-- Name: audit_log_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres

--


ALTER SEQUENCE public.audit_log_id_seq OWNED BY public.audit_log.id;


--

-- Name: data_versions; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public.data_versions (

    pk_version integer NOT NULL,

    table_name character varying(100) NOT NULL,

    record_id integer NOT NULL,

    action character varying(20) NOT NULL,

    data_before json,

    data_after json,

    changed_by integer,
```

```
    changed_at timestamp without time zone,
    created_at timestamp without time zone
);


ALTER TABLE public.data_versions OWNER TO postgres;


--
-- Name: data_versions_pk_version_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--


CREATE SEQUENCE public.data_versions_pk_version_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.data_versions_pk_version_seq OWNER TO postgres;


--
-- Name: data_versions_pk_version_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--
```

```
ALTER SEQUENCE public.data_versions_pk_version_seq OWNED BY
public.data_versions.pk_version;


--

-- Name: import_history; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public.import_history (

    id integer NOT NULL,

    filename character varying(255) NOT NULL,

    file_type character varying(10) NOT NULL,

    imported_by integer,

    imported_at timestamp without time zone DEFAULT
CURRENT_TIMESTAMP,

    record_count integer,

    status character varying(20) DEFAULT 'completed'::character varying,

    errors text

);


ALTER TABLE public.import_history OWNER TO postgres;


--

-- Name: import_history_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres

--
```

```
CREATE SEQUENCE public.import_history_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.import_history_id_seq OWNER TO postgres;


--
-- Name: import_history_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres
--


ALTER SEQUENCE public.import_history_id_seq OWNED BY public.import_history.id;


--
-- Name: school_versions; Type: TABLE; Schema: public; Owner: postgres
--


CREATE TABLE public.school_versions (
    pk_version integer NOT NULL,
```

```
    pk_school integer NOT NULL,
    version_number integer NOT NULL,
    action character varying(50) NOT NULL,
    old_data text,
    new_data text,
    changed_by integer,
    changed_at timestamp without time zone
);

ALTER TABLE public.school_versions OWNER TO postgres;

--
-- Name: school_versions_pk_version_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.school_versions_pk_version_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER SEQUENCE public.school_versions_pk_version_seq OWNER TO
postgres;


--

-- Name: school_versions_pk_version_seq; Type: SEQUENCE OWNED BY;
Schema: public; Owner: postgres

--


ALTER SEQUENCE public.school_versions_pk_version_seq OWNED BY
public.school_versions.pk_version;


--

-- Name: users; Type: TABLE; Schema: public; Owner: postgres

--


CREATE TABLE public.users (

    id integer NOT NULL,

    username character varying(64) NOT NULL,

    email character varying(120) NOT NULL,

    password_hash character varying(256),

    role integer DEFAULT 1,

    is_active boolean DEFAULT true,

    created_at timestamp without time zone DEFAULT
CURRENT_TIMESTAMP,

    last_login timestamp without time zone,

    gosuslugi_id character varying(100),
```

```
    gosuslugi_data text
);


ALTER TABLE public.users OWNER TO postgres;


--
-- Name: users_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--


CREATE SEQUENCE public.users_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER SEQUENCE public.users_id_seq OWNER TO postgres;


--
-- Name: users_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
Owner: postgres
--


ALTER SEQUENCE public.users_id_seq OWNED BY public.users.id;
```

--

-- Name: District PK_District; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."District" ALTER COLUMN "PK_District" SET DEFAULT nextval('public."District_PK_District_seq"'::regclass);

--

-- Name: Education_Program PK_Education_Program; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Education_Program" ALTER COLUMN "PK_Education_Program" SET DEFAULT nextval('public."Education_Program_PK_Education_Program_seq"'::regclass);

--

-- Name: Employee PK_Employee; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Employee" ALTER COLUMN "PK_Employee" SET DEFAULT nextval('public."Employee_PK_Employee_seq"'::regclass);

--

-- Name: Infrastructure PK_Infrastructure; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Infrastructure" ALTER COLUMN "PK_Infrastructure" SET DEFAULT nextval('public."Infrastructure_PK_Infrastructure_seq"'::regclass);

--

-- Name: Inspection PK_Inspection; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Inspection" ALTER COLUMN "PK_Inspection" SET DEFAULT nextval('public."Inspection_PK_Inspection_seq"'::regclass);

--

-- Name: Review PK_Review; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Review" ALTER COLUMN "PK_Review" SET DEFAULT nextval('public."Review_PK_Review_seq"'::regclass);

--

-- Name: School PK_School; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."School" ALTER COLUMN "PK_School" SET DEFAULT nextval('public."School_PK_School_seq"'::regclass);

--

-- Name: Settlement PK_Settlement; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Settlement" ALTER COLUMN "PK_Settlement" SET DEFAULT nextval('public."Settlement_PK_Settlement_seq"'::regclass);

--

-- Name: Specialization PK_Specialization; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Specialization" ALTER COLUMN "PK_Specialization" SET DEFAULT nextval('public."Specialization_PK_Specialization_seq"'::regclass);

--

-- Name: Subject PK_Subject; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Subject" ALTER COLUMN "PK_Subject" SET DEFAULT nextval('public."Subject_PK_Subject_seq"'::regclass);

--

-- Name: Type_of_School PK_Type_of_School; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."Type_of_School" ALTER COLUMN "PK_Type_of_School" SET DEFAULT nextval('public."Type_of_School_PK_Type_of_School_seq"'::regclass);

--

-- Name: audit_log id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.audit_log ALTER COLUMN id SET DEFAULT nextval('public.audit_log_id_seq'::regclass);

--

-- Name: data_versions pk_version; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.data_versions ALTER COLUMN pk_version SET DEFAULT nextval('public.data_versions_pk_version_seq'::regclass);

--

-- Name: import_history id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.import_history ALTER COLUMN id SET DEFAULT nextval('public.import_history_id_seq'::regclass);

--

-- Name: school_versions pk_version; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.school_versions ALTER COLUMN pk_version SET DEFAULT nextval('public.school_versions_pk_version_seq'::regclass);

--

-- Name: users id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.users ALTER COLUMN id SET DEFAULT nextval('public.users_id_seq'::regclass);

Приложение В
Исходный код программы

github: https://github.com/yehoto/Coursework_Schools_in_the_region

static/css/style.css

```
/* Основные стили */
:root {
    --primary-color: #0d6efd;
    --secondary-color: #6c757d;
    --success-color: #198754;
    --info-color: #0dcaf0;
    --warning-color: #ffc107;
    --danger-color: #dc3545;
    --light-color: #f8f9fa;
    --dark-color: #212529;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    padding-top: 70px;
    background-color: #f8f9fa;
}

/* Навигация */
.navbar {
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.navbar-brand {
    font-weight: 600;
}

/* Карточки */
```

```css
.card {
    border: none;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    transition: transform 0.2s, box-shadow 0.2s;
    border-radius: 10px;
    overflow: hidden;
}

.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0,0,0,0.1);
}

.card-header {
    border-bottom: none;
    font-weight: 600;
}

/* Школы */
.school-card {
    height: 100%;
}

.school-card .card-title {
    color: var(--dark-color);
    font-weight: 600;
}

.school-card .card-subtitle {
    font-size: 0.9rem;
}
```

```
/* Рейтинг */

.rating-stars {

    color: var(--warning-color);

}


/* Кнопки */

.btn {

    border-radius: 6px;

    font-weight: 500;

    padding: 0.5rem 1.5rem;

}


.btn-primary {

    background-color: var(--primary-color);

    border-color: var(--primary-color);

}


.btn-primary:hover {

    background-color: #0b5ed7;

    border-color: #0b5ed7;

}


.btn-outline-primary {

    color: var(--primary-color);

    border-color: var(--primary-color);

}


.btn-outline-primary:hover {

    background-color: var(--primary-color);

    border-color: var(--primary-color);

}
```

```css
/* Формы */
.form-control, .form-select {
    border-radius: 6px;
    border: 1px solid #dee2e6;
    padding: 0.5rem 1rem;
}

.form-control:focus, .form-select:focus {
    border-color: var(--primary-color);
    box-shadow: 0 0 0 0.25rem rgba(13, 110, 253, 0.25);
}

.form-label {
    font-weight: 500;
    margin-bottom: 0.5rem;
}

/* Подвал */
.footer {
    margin-top: 3rem;
    box-shadow: 0 -2px 10px rgba(0,0,0,0.05);
}

.footer h5, .footer h6 {
    font-weight: 600;
}

.footer ul li {
    margin-bottom: 0.5rem;
}

/* Пагинация */
```

```css
.pagination .page-link {
    border-radius: 6px;
    margin: 0 3px;
    border: none;
    color: var(--secondary-color);
}

.pagination .page-link:hover {
    background-color: var(--light-color);
    color: var(--primary-color);
}

.pagination .active .page-link {
    background-color: var(--primary-color);
    border-color: var(--primary-color);
}

/* Алерты */
.alert {
    border-radius: 8px;
    border: none;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
}

.alert-dismissible .btn-close {
    padding: 1rem;
}

/* Утилиты */
.text-small {
    font-size: 0.875rem;
}
```

```css
.text-xsmall {
    font-size: 0.75rem;
}

/* Адаптивность */
@media (max-width: 768px) {
    body {
        padding-top: 60px;
    }

    .navbar-brand span {
        display: none;
    }

    .card {
        margin-bottom: 1rem;
    }

    .btn {
        padding: 0.375rem 1rem;
    }
}

/* Анимации */
@keyframes fadeIn {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

.fade-in {
    animation: fadeIn 0.3s ease-out;
```

```
}

/* Бейджи */
.badge {
    font-weight: 500;
    padding: 0.35em 0.65em;
    border-radius: 6px;
}

/* Таблицы */
.table {
    background-color: white;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
}

.table thead th {
    background-color: var(--light-color);
    border-bottom: 2px solid #dee2e6;
    font-weight: 600;
    color: var(--dark-color);
}

.table tbody tr:hover {
    background-color: rgba(0,0,0,0.02);
}

/* Загрузка */
.loading {
    display: inline-block;
    width: 20px;
```

```
    height: 20px;
    border: 3px solid rgba(0,0,0,0.1);
    border-radius: 50%;
    border-top-color: var(--primary-color);
    animation: spin 1s ease-in-out infinite;
}

@keyframes spin {
    to { transform: rotate(360deg); }
}
```

static/js/main.js

```
// Основной JavaScript файл

// Инициализация при загрузке страницы
document.addEventListener('DOMContentLoaded', function() {
    // Инициализация всех компонентов
    initComponents();

    // Обработка форм
    initForms();

    // Динамическая загрузка данных
    initDynamicLoading();
});

// Инициализация компонентов
function initComponents() {
    // Всплывающие подсказки
    var tooltipTriggerList =
[].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'));
    var tooltipList = tooltipTriggerList.map(function
(tooltipTriggerEl) {
        return new bootstrap.Tooltip(tooltipTriggerEl);
```

```
    });

    // Всплывающие окна
    var popoverTriggerList =
[].slice.call(document.querySelectorAll('[data-bs-toggle="popover"]'));
    var popoverList = popoverTriggerList.map(function
(popoverTriggerEl) {
        return new bootstrap.Popover(popoverTriggerEl);
    });

    // Подтверждение действий
    var confirmActions = document.querySelectorAll('.confirm-action');
    confirmActions.forEach(function(element) {
        element.addEventListener('click', function(e) {
            if (!confirm('Вы уверены?')) {
                e.preventDefault();
                return false;
            }
        });
    });
}

// Обработка форм
function initForms() {
    // Динамическое обновление населенных пунктов при выборе района
    var districtSelect = document.getElementById('district-select');
    if (districtSelect) {
        districtSelect.addEventListener('change', function() {
            var districtId = this.value;
            var settlementSelect =
document.getElementById('settlement-select');

            if (districtId) {
```

```
                fetch(`/api/districts/${districtId}/settlements`)
                    .then(response => response.json())
                    .then(data => {
                        settlementSelect.innerHTML = '<option
value="">Выберите населенный пункт</option>';
                        data.forEach(function(settlement) {
                            var option =
document.createElement('option');
                            option.value = settlement.id;
                            option.textContent = settlement.type + ' '
+ settlement.name;
                            settlementSelect.appendChild(option);
                        });
                    })
                    .catch(error => {
                        console.error('Ошибка при загрузке населенных
пунктов:', error);
                    });
            }
        });
    }


    // Валидация форм
    var forms = document.querySelectorAll('.needs-validation');
    forms.forEach(function(form) {
        form.addEventListener('submit', function(event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        });
    });
```

```
    // Подсчет символов в текстовых полях
    var textAreas =
document.querySelectorAll('textarea[data-max-length]');
    textAreas.forEach(function(textarea) {
        var maxLength = textarea.getAttribute('data-max-length');
        var counter = document.createElement('div');
        counter.className = 'form-text text-end small';
        counter.innerHTML = `<span class="char-count">0</span> /
${maxLength}`;

        textarea.parentNode.appendChild(counter);

        textarea.addEventListener('input', function() {
            var length = this.value.length;
            var charCount =
this.parentNode.querySelector('.char-count');
            charCount.textContent = length;

            if (length > maxLength) {
                charCount.classList.add('text-danger');
            } else {
                charCount.classList.remove('text-danger');
            }
        });
    });
}

// Динамическая загрузка данных
function initDynamicLoading() {
    // Ленивая загрузка изображений
    var lazyImages = document.querySelectorAll('img[data-src]');

    if ('IntersectionObserver' in window) {
```

```
        var imageObserver = new IntersectionObserver(function(entries,
observer) {
            entries.forEach(function(entry) {
                if (entry.isIntersecting) {
                    var img = entry.target;
                    img.src = img.getAttribute('data-src');
                    img.removeAttribute('data-src');
                    imageObserver.unobserve(img);
                }
            });
        });

        lazyImages.forEach(function(img) {
            imageObserver.observe(img);
        });
    } else {
        // Fallback для старых браузеров
        lazyImages.forEach(function(img) {
            img.src = img.getAttribute('data-src');
        });
    }

    // Бесконечная прокрутка (если есть)
    var infiniteScrollContainer =
document.querySelector('.infinite-scroll');
    if (infiniteScrollContainer) {
        window.addEventListener('scroll', function() {
            if (window.innerHeight + window.scrollY >=
document.body.offsetHeight - 500) {
                loadMoreContent();
            }
        });
    }
```

```
}


// Загрузка дополнительного контента
function loadMoreContent() {
    var nextPage = document.querySelector('.pagination
.active').nextElementSibling;
    if (!nextPage || nextPage.classList.contains('disabled')) {
        return;
    }

    var url = nextPage.querySelector('a').href;
    var loadingIndicator =
document.getElementById('loading-indicator');

    if (loadingIndicator) {
        loadingIndicator.style.display = 'block';
    }

    fetch(url)
        .then(response => response.text())
        .then(html => {
            var parser = new DOMParser();
            var doc = parser.parseFromString(html, 'text/html');
            var newItems =
doc.querySelector('.schools-container').innerHTML;

            document.querySelector('.schools-container').innerHTML +=
newItems;

            if (loadingIndicator) {
                loadingIndicator.style.display = 'none';
            }

            // Обновляем URL без перезагрузки страницы
```

```
            window.history.pushState({}, '', url);


            // Повторно инициализируем компоненты для новых элементов
            initComponents();
        })
        .catch(error => {
            console.error('Ошибка при загрузке контента:', error);
            if (loadingIndicator) {
                loadingIndicator.style.display = 'none';
            }
        });
}


// Вспомогательные функции
function showToast(message, type = 'info') {
    var toastContainer = document.getElementById('toast-container');
    if (!toastContainer) {
        toastContainer = document.createElement('div');
        toastContainer.id = 'toast-container';
        toastContainer.className = 'toast-container position-fixed
bottom-0 end-0 p-3';
        document.body.appendChild(toastContainer);
    }


    var toastId = 'toast-' + Date.now();
    var toastHtml = `
        <div id="${toastId}" class="toast" role="alert"
aria-live="assertive" aria-atomic="true">
            <div class="toast-header bg-${type} text-white">
                <strong class="me-auto">Уведомление</strong>
                <button type="button" class="btn-close btn-close-white"
data-bs-dismiss="toast"></button>
            </div>
```

```
            <div class="toast-body">
                ${message}
            </div>
        </div>
    `;

    toastContainer.insertAdjacentHTML('beforeend', toastHtml);

    var toastElement = document.getElementById(toastId);
    var toast = new bootstrap.Toast(toastElement);
    toast.show();

    // Удаляем toast после скрытия
    toastElement.addEventListener('hidden.bs.toast', function () {
        toastElement.remove();
    });
}

// AJAX запросы с обработкой ошибок
function ajaxRequest(url, options = {}) {
    const defaults = {
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
            'X-Requested-With': 'XMLHttpRequest'
        }
    };

    const config = { ...defaults, ...options };

    return fetch(url, config)
        .then(response => {
```

```
            if (!response.ok) {
                throw new Error(`HTTP error! status:
${response.status}`);
            }
            return response.json();
        })
        .catch(error => {
            console.error('AJAX request failed:', error);
            showToast('Произошла ошибка при загрузке данных',
'danger');
            throw error;
        });
}

// Форматирование чисел
function formatNumber(number) {
    return new Intl.NumberFormat('ru-RU').format(number);
}

// Форматирование даты
function formatDate(dateString) {
    const date = new Date(dateString);
    return date.toLocaleDateString('ru-RU', {
        year: 'numeric',
        month: 'long',
        day: 'numeric'
    });
}

// Копирование в буфер обмена
function copyToClipboard(text) {
    navigator.clipboard.writeText(text)
        .then(() => {
```

```
            showToast('Скопировано в буфер обмена', 'success');
        })
        .catch(err => {
            console.error('Ошибка при копировании:', err);
            showToast('Ошибка при копировании', 'danger');
        });
}

// Экспорт модулей (если используется модульная система)
if (typeof module !== 'undefined' && module.exports) {
    module.exports = {
        initComponents,
        initForms,
        initDynamicLoading,
        showToast,
        ajaxRequest,
        formatNumber,
        formatDate,
        copyToClipboard
    };
}
```

templates/admin/add_inspection.html

```
{% extends "base.html" %}

{% block title %}Добавить проверку - Реестр школ Алтайского края{% endblock %}

{% block content %}
<div class="container">
    <div class="row justify-content-center">
        <div class="col-lg-8">
            <div class="card">
```

```
                <div class="card-header bg-primary text-white">
                    <h4 class="mb-0"><i class="bi bi-plus-circle"></i>
Добавить проверку Рособрнадзора</h4>
                </div>
                <div class="card-body">
                    <form method="POST" action="{{
url_for('add_inspection') }}" class="needs-validation" novalidate>
                        {{ form.hidden_tag() }}

                        <div class="row">
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.date.label(class="form-label") }}
                                    {{ form.date(class="form-control",
required=true) }}
                                    {% if form.date.errors %}
                                    <div class="invalid-feedback
d-block">
                                        {{ form.date.errors[0] }}
                                    </div>
                                    {% endif %}
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.prescription_number.label(class="form-label") }}
                                    {{
form.prescription_number(class="form-control", placeholder="Например:
01-01-2024/123") }}
                                    {% if
form.prescription_number.errors %}
                                    <div class="invalid-feedback
d-block">
```

```
                                    {{
form.prescription_number.errors[0] }}
                                </div>
                                {% endif %}
                            </div>
                        </div>

                        <div class="mb-3">
                            {{ form.school_id.label(class="form-label")
}}
                            {{ form.school_id(class="form-select") }}
                        </div>

                        <div class="mb-3">
                            {{ form.result.label(class="form-label") }}
                            {{ form.result(class="form-control",
rows="3", placeholder="Опишите результаты проверки...") }}
                        </div>

                        <div class="mb-3">
                            <div class="form-check">
                                {{
form.has_violations(class="form-check-input") }}
                                {{
form.has_violations.label(class="form-check-label") }}
                            </div>
                        </div>

                        <div class="mb-3">
                            {{
form.violation_type.label(class="form-label") }}
                            {{
form.violation_type(class="form-control", placeholder="Например:
СанПиН, пожарная безопасность") }}
```

```
                    </div>


                    <div class="row">
                        <div class="col-md-6">
                            <div class="mb-3">
                                <div class="form-check">
                                    {{
form.is_resolved(class="form-check-input") }}
                                    {{
form.is_resolved.label(class="form-check-label") }}
                                </div>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="mb-3">
                                {{
form.resolution_date.label(class="form-label") }}
                                {{
form.resolution_date(class="form-control") }}
                            </div>
                        </div>
                    </div>


                    <div class="mb-3">
                        {{
form.description.label(class="form-label") }}
                        {{ form.description(class="form-control",
rows="3", placeholder="Дополнительное описание проверки...") }}
                    </div>


                    <div class="mt-4 d-flex
justify-content-between">
                        <a href="{{ url_for('admin_inspections')
}}" class="btn btn-secondary">
                            <i class="bi bi-arrow-left"></i> Отмена
```

```
                        </a>
                        {{ form.submit(class="btn btn-primary") }}
                    </div>
                </form>
            </div>
        </div>

        <div class="card mt-4">
            <div class="card-header bg-info text-white">
                <h6 class="mb-0"><i class="bi bi-info-circle"></i> Подсказки</h6>
            </div>
            <div class="card-body">
                <ul class="mb-0">
                    <li>Все поля, помеченные <strong>*</strong>, обязательны для заполнения</li>
                    <li>Номер предписания должен быть уникальным для каждой проверки</li>
                    <li>Отметьте "Нарушения выявлены", если в ходе проверки были обнаружены нарушения</li>
                    <li>Укажите "Дата устранения" и отметьте "Нарушения устранены" после исправления нарушений</li>
                </ul>
            </div>
        </div>
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        // Валидация форм
        var forms = document.querySelectorAll('.needs-validation');
        Array.prototype.slice.call(forms).forEach(function(form) {
```

```
            form.addEventListener('submit', function(event) {
                if (!form.checkValidity()) {
                    event.preventDefault();
                    event.stopPropagation();
                }
                form.classList.add('was-validated');
            });
        });

        // Показать/скрыть поля в зависимости от нарушения
        var hasViolationsCheckbox =
document.getElementById('has_violations');
        var violationTypeField =
document.getElementById('violation_type');
        var isResolvedCheckbox =
document.getElementById('is_resolved');
        var resolutionDateField =
document.getElementById('resolution_date');

        function toggleViolationFields() {
            if (hasViolationsCheckbox.checked) {
                violationTypeField.parentElement.style.display =
'block';
                isResolvedCheckbox.parentElement.style.display =
'block';
                if (isResolvedCheckbox.checked) {
                    resolutionDateField.parentElement.style.display =
'block';
                } else {
                    resolutionDateField.parentElement.style.display =
'none';
                }
            } else {
                violationTypeField.parentElement.style.display =
'none';
```

```
                isResolvedCheckbox.parentElement.style.display =
'none';

                resolutionDateField.parentElement.style.display =
'none';

            }

        }


        hasViolationsCheckbox.addEventListener('change',
toggleViolationFields);

        isResolvedCheckbox.addEventListener('change',
toggleViolationFields);


        // Инициализация при загрузке

        toggleViolationFields();

    });

</script>

{% endblock %}
```

templates/admin/add_school.html

```
{% extends "base.html" %}


{% block title %}Добавить школу - Реестр школ Алтайского края{%
endblock %}


{% block content %}

<div class="container">

    <div class="row justify-content-center">

        <div class="col-lg-10">

            <div class="card">

                <div class="card-header bg-success text-white">

                    <h4 class="mb-0"><i class="bi bi-plus-circle"></i>
Добавить новую школу</h4>

                </div>

                <div class="card-body">

                    <form method="POST" action="{{
url_for('add_school') }}" class="needs-validation" novalidate>
```

```
                    {{ form.hidden_tag() }}

                    <div class="row">
                        <!-- Основная информация -->
                        <div class="col-md-6">
                            <h5 class="mb-3 border-bottom
pb-2">Основная информация</h5>

                            <div class="mb-3">
                                {{
form.official_name.label(class="form-label") }}
                                {{
form.official_name(class="form-control", placeholder="Официальное
название школы") }}
                                {% if form.official_name.errors %}
                                <div class="invalid-feedback
d-block">
                                    {{ form.official_name.errors[0]
}}
                                </div>
                                {% endif %}
                            </div>

                            <div class="mb-3">
                                {{
form.legal_address.label(class="form-label") }}
                                {{
form.legal_address(class="form-control", rows="3",
placeholder="Юридический адрес") }}
                                {% if form.legal_address.errors %}
                                <div class="invalid-feedback
d-block">
                                    {{ form.legal_address.errors[0]
}}
                                </div>
                                {% endif %}
```

```
                                </div>


                                <div class="row mb-3">
                                    <div class="col-md-6">
                                        {{
form.type_of_school.label(class="form-label") }}
                                        {{
form.type_of_school(class="form-select") }}
                                    </div>
                                    <div class="col-md-6">
                                        {{
form.settlement.label(class="form-label") }}
                                        {{
form.settlement(class="form-select") }}
                                    </div>
                                </div>
                            </div>


                            <!-- Контактная информация -->
                            <div class="col-md-6">
                                <h5 class="mb-3 border-bottom
pb-2">Контактная информация</h5>


                                <div class="mb-3">
                                    {{
form.phone.label(class="form-label") }}
                                    {{ form.phone(class="form-control",
placeholder="+7 (3852) 123456") }}
                                    {% if form.phone.errors %}
                                    <div class="invalid-feedback
d-block">
                                        {{ form.phone.errors[0] }}
                                    </div>
                                    {% endif %}
                                </div>
```

```
                                <div class="mb-3">
                                    {{
form.email.label(class="form-label") }}
                                    {{ form.email(class="form-control",
placeholder="school@example.com") }}
                                </div>

                                <div class="mb-3">
                                    {{
form.website.label(class="form-label") }}
                                    {{
form.website(class="form-control", placeholder="http://example.com") }}
                                </div>

                                <div class="row mb-3">
                                    <div class="col-md-6">
                                        {{
form.founding_date.label(class="form-label") }}
                                        {{
form.founding_date(class="form-control") }}
                                    </div>
                                    <div class="col-md-6">
                                        {{
form.number_of_students.label(class="form-label") }}
                                        {{
form.number_of_students(class="form-control", placeholder="500") }}
                                    </div>
                                </div>
                            </div>
                        </div>

                        <!-- Дополнительная информация -->
                        <div class="row mt-3">
                            <div class="col-md-6">
```

```
                                <h5 class="mb-3 border-bottom
pb-2">Дополнительно</h5>

                                <div class="mb-3">
                                    {{
form.infrastructure.label(class="form-label") }}
                                    {{
form.infrastructure(class="form-select", multiple=True, size="5") }}
                                    <small
class="text-muted">Удерживайте Ctrl для выбора нескольких
вариантов</small>
                                </div>

                                <div class="mb-3">
                                    {{
form.specializations.label(class="form-label") }}
                                    {{
form.specializations(class="form-select", multiple=True, size="5") }}
                                </div>
                            </div>

                            <div class="col-md-6">
                                <h5 class="mb-3 border-bottom
pb-2">Документы</h5>

                                <div class="mb-3">
                                    {{
form.license.label(class="form-label") }}
                                    {{
form.license(class="form-control", rows="3", placeholder="Информация о
лицензии") }}
                                </div>

                                <div class="mb-3">
                                    {{
form.accreditation.label(class="form-label") }}
```

```
                                {{
form.accreditation(class="form-control", rows="3",
placeholder="Информация об аккредитации") }}
                            </div>
                        </div>
                    </div>

                    <div class="mt-4 d-flex
justify-content-between">
                        <a href="{{ url_for('index') }}" class="btn
btn-secondary">
                            <i class="bi bi-arrow-left"></i> Отмена
                        </a>  <!-- Закрывающий тег ПЕРЕД
содержимым! -->
                        {{ form.submit(class="btn btn-success") }}
                    </div>
                </form>
            </div>
        </div>

        <!-- Подсказки -->
        <div class="card mt-4">
            <div class="card-header bg-info text-white">
                <h6 class="mb-0"><i class="bi bi-info-circle"></i>
Подсказки</h6>
            </div>
            <div class="card-body">
                <ul class="mb-0">
                    <li>Все поля, помеченные <strong>*</strong>,
обязательны для заполнения</li>
                    <li>Для выбора нескольких вариантов
инфраструктуры или специализаций удерживайте клавишу Ctrl</li>
                    <li>Дата основания должна быть в формате
ГГГГ-ММ-ДД (например, 1950-09-01)</li>
                    <li>После добавления школы вы сможете добавить
сотрудников и программы</li>
```

```
                    </ul>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
document.addEventListener('DOMContentLoaded', function() {
    // Валидация форм
    var forms = document.querySelectorAll('.needs-validation');
    Array.prototype.slice.call(forms).forEach(function(form) {
        form.addEventListener('submit', function(event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        });
    });

    // Подсказка по дате
    var dateField = document.getElementById('founding_date');
    if (dateField) {
        dateField.addEventListener('focus', function() {
            if (!this.value) {
                this.type = 'date';
            }
        });
        dateField.addEventListener('blur', function() {
            if (!this.value) {
                this.type = 'text';
```

```
            }

        });

    }

});

</script>

{% endblock %}
```

templates/admin/dashboard.html

```
{% extends "base.html" %}


{% block title %}Панель управления - Реестр школ Алтайского края{%
endblock %}


{% block content %}

<div class="container-fluid">

    <div class="d-flex justify-content-between align-items-center
mb-4">

        <h1><i class="bi bi-speedometer2"></i> Панель управления</h1>

        <div class="btn-group" role="group">

            {% if current_user.has_role('region_admin') %}

            <a href="{{ url_for('add_school') }}" class="btn
btn-success">

                <i class="bi bi-plus-circle"></i> Добавить школу

            </a>

            {% endif %}

            <a href="{{ url_for('admin_inspections') }}" class="btn
btn-outline-primary">

                <i class="bi bi-clipboard-check"></i> Управление
проверками

            </a>

        </div>

    </div>


    <!-- Статистика -->

    <div class="row mb-4">
```

```
        <div class="col-md-4">
            <div class="card bg-primary text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего школ</h6>
                    <h2 class="mb-0">{{ stats.total_schools }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card bg-info text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего учащихся</h6>
                    <h2 class="mb-0">{{ stats.total_students }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card bg-warning text-dark">
                <div class="card-body text-center">
                    <h6 class="card-title">Отзывов на модерации</h6>
                    <h2 class="mb-0">{{ pending_reviews }}</h2>
                </div>
            </div>
        </div>
    </div>

    <!-- Панель управления -->
    <div class="row mb-4">
        <div class="col-md-12">
            <div class="card">
                <div class="card-header bg-light">
                    <h5 class="mb-0"><i class="bi bi-gear"></i> Панель
управления</h5>
```

```
                </div>
                <div class="card-body">
                    <div class="row g-3">
                        <div class="col-md-4">
                            <a href="{{ url_for('admin_reviews') }}"
class="btn btn-outline-warning w-100">
                                <i class="bi bi-chat-text"></i>
Модерация отзывов
                                {% if pending_reviews > 0 %}
                                <span class="badge bg-danger ms-1">{{
pending_reviews }}</span>
                                {% endif %}
                            </a>
                        </div>
                        <div class="col-md-4">
                            <a href="{{ url_for('admin_inspections')
}}" class="btn btn-outline-primary w-100">
                                <i class="bi bi-clipboard-check"></i>
Управление проверками
                            </a>
                        </div>
                        <div class="col-md-4">
                            <a href="{{ url_for('create_backup_route')
}}" class="btn btn-outline-success w-100">
                                <i class="bi bi-download"></i> Создать
бэкап (dump.sql)
                            </a>
                        </div>
                    </div>
                </div>
            </div>
        </div>

    <!-- Последние действия -->
```

```
    <div class="row">
        <div class="col-md-12">
            <div class="card">
                <div class="card-header">
                    <h5 class="mb-0"><i class="bi
bi-clock-history"></i> Последние действия в системе</h5>
                </div>
                <div class="card-body">
                    {% if recent_audits %}
                    <div class="table-responsive">
                        <table class="table table-hover">
                            <thead>
                                <tr>
                                    <th>Дата/Время</th>
                                    <th>Пользователь</th>
                                    <th>Действие</th>
                                    <th>Объект</th>
                                    <th>Детали</th>
                                </tr>
                            </thead>
                            <tbody>
                                {% for audit in recent_audits %}
                                <tr>
                                    <td>
                                        <small>
                                            {% if audit.timestamp %}
                                            {{
audit.timestamp.strftime('%d.%m.%Y %H:%M') }}
                                            {% else %}
                                            <span
class="text-muted">Нет даты</span>
                                            {% endif %}
                                        </small>
```

```
                                    </td>
                                    <td>
                                        {% if audit.user_id %}
                                            {% set user =
get_user_by_id(audit.user_id) %}
                                            {% if user %}
                                            <strong>{{ user.username
}}</strong>
                                            {% else %}
                                            <span
class="text-muted">ID: {{ audit.user_id }}</span>
                                            {% endif %}
                                        {% else %}
                                            <span
class="text-muted">Система</span>
                                        {% endif %}
                                    </td>
                                    <td>
                                        {% if audit.action == 'create'
%}
                                        <span class="badge
bg-success">Создание</span>
                                        {% elif audit.action ==
'update' %}
                                        <span class="badge bg-warning
text-dark">Изменение</span>
                                        {% elif audit.action ==
'delete' %}
                                        <span class="badge
bg-danger">Удаление</span>
                                        {% else %}
                                        <span class="badge
bg-secondary">{{ audit.action|upper }}</span>
                                        {% endif %}
                                    </td>
                                    <td>
```

```
                                        <span class="fw-medium">{{
audit.table_name }}</span>
                                        {% if audit.record_id %}
                                        <span class="text-muted
ms-1">#{{ audit.record_id }}</span>
                                        {% endif %}
                                    </td>
                                    <td>
                                        {% if audit.table_name ==
'School' and audit.record_id %}
                                        <div class="btn-group
btn-group-sm" role="group">
                                            <a href="{{
url_for('school_detail', school_id=audit.record_id) }}"
                                               class="btn
btn-outline-primary">
                                                <i class="bi
bi-eye"></i>
                                            </a>
                                            {% if audit.action ==
'update' %}
                                            <a href="{{
url_for('school_history', school_id=audit.record_id) }}"
                                               class="btn
btn-outline-info">
                                                <i class="bi
bi-clock-history"></i>
                                            </a>
                                            {% endif %}
                                        </div>
                                        {% endif %}
                                    </td>
                                </tr>
                                {% endfor %}
                            </tbody>
                        </table>
```

```
                    </div>
                    {% else %}
                    <div class="text-center py-4">
                        <i class="bi bi-activity display-4
text-muted"></i>
                        <p class="mt-3 text-muted">Нет записей о
действиях</p>
                    </div>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

templates/admin/edit_inspection.html

```
{% extends "base.html" %}

{% block title %}Редактировать проверку - Реестр школ Алтайского края{%
endblock %}

{% block content %}
<div class="container">
    <div class="row justify-content-center">
        <div class="col-lg-8">
            <div class="card">
                <div class="card-header bg-warning text-dark">
                    <h4 class="mb-0"><i class="bi bi-pencil"></i>
Редактировать проверку</h4>
                    <p class="mb-0 small">Номер предписания: {{
inspection.Prescription_Number }}</p>
                </div>
                <div class="card-body">
```

```
                    <form method="POST" action="{{
url_for('edit_inspection', inspection_id=inspection.PK_Inspection) }}"
class="needs-validation" novalidate>

                        {{ form.hidden_tag() }}


                        <div class="alert alert-info">

                            <i class="bi bi-info-circle"></i> Вы
редактируете проверку от {{ inspection.Date.strftime('%d.%m.%Y') }}

                        </div>


                        <div class="row">

                            <div class="col-md-6">

                                <div class="mb-3">

                                    {{
form.date.label(class="form-label") }}

                                    {{ form.date(class="form-control",
required=true) }}

                                </div>

                            </div>

                            <div class="col-md-6">

                                <div class="mb-3">

                                    {{
form.prescription_number.label(class="form-label") }}

                                    {{
form.prescription_number(class="form-control") }}

                                </div>

                            </div>

                        </div>


                        <div class="mb-3">

                            {{ form.school_id.label(class="form-label")
}}

                            {{ form.school_id(class="form-select") }}

                        </div>
```

```
                        <div class="mb-3">
                            {{ form.result.label(class="form-label") }}
                            {{ form.result(class="form-control",
rows="3") }}
                        </div>

                        <div class="mb-3">
                            <div class="form-check">
                                {{
form.has_violations(class="form-check-input") }}
                                {{
form.has_violations.label(class="form-check-label") }}
                            </div>
                        </div>

                        <div class="mb-3">
                            {{
form.violation_type.label(class="form-label") }}
                            {{
form.violation_type(class="form-control") }}
                        </div>

                        <div class="row">
                            <div class="col-md-6">
                                <div class="mb-3">
                                    <div class="form-check">
                                        {{
form.is_resolved(class="form-check-input") }}
                                        {{
form.is_resolved.label(class="form-check-label") }}
                                    </div>
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
```

```
                                    {{
form.resolution_date.label(class="form-label") }}

                                    {{
form.resolution_date(class="form-control") }}

                                </div>

                            </div>

                        </div>



                        <div class="mb-3">

                            {{
form.description.label(class="form-label") }}

                            {{ form.description(class="form-control",
rows="3") }}

                        </div>



                        <div class="mt-4 d-flex
justify-content-between">

                            <a href="{{ url_for('admin_inspections')
}}" class="btn btn-secondary">

                                <i class="bi bi-arrow-left"></i> Отмена

                            </a>

                            {{ form.submit(class="btn btn-warning") }}

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>



<script>

    document.addEventListener('DOMContentLoaded', function() {

        var forms = document.querySelectorAll('.needs-validation');

        Array.prototype.slice.call(forms).forEach(function(form) {

            form.addEventListener('submit', function(event) {
```

```
                if (!form.checkValidity()) {
                    event.preventDefault();
                    event.stopPropagation();
                }
                form.classList.add('was-validated');
            });
        });
    });
</script>
{% endblock %}
```

templates/admin/edit_school.html

```
{% extends "base.html" %}

{% block title %}Редактировать {{ school.Official_Name }} - Реестр школ Алтайского края{% endblock %}

{% block content %}
<div class="container">
    <div class="row justify-content-center">
        <div class="col-lg-10">
            <div class="card">
                <div class="card-header bg-warning text-dark">
                    <h4 class="mb-0"><i class="bi bi-pencil"></i> Редактировать школу</h4>
                    <p class="mb-0 small">ID: {{ school.PK_School }}</p>
                </div>
                <div class="card-body">
                    <form method="POST" action="{{ url_for('edit_school', school_id=school.PK_School) }}" class="needs-validation" novalidate>
                        {{ form.hidden_tag() }}

                        <div class="alert alert-info">
```

```
                        <i class="bi bi-info-circle"></i> Вы
редактируете: <strong>{{ school.Official_Name }}</strong>

                    </div>



                    <!-- Основная информация -->

                    <h5 class="mb-3 border-bottom pb-2">Основная
информация</h5>

                    <div class="row">

                        <div class="col-md-6">

                            <div class="mb-3">

                                {{
form.official_name.label(class="form-label") }}

                                {{
form.official_name(class="form-control", placeholder="Официальное
название школы") }}

                            </div>

                        </div>

                        <div class="col-md-6">

                            <div class="mb-3">

                                {{
form.phone.label(class="form-label") }}

                                {{ form.phone(class="form-control",
placeholder="+7 (3852) 123456") }}

                            </div>

                        </div>

                    </div>



                    <div class="mb-3">

                        {{
form.legal_address.label(class="form-label") }}

                        {{ form.legal_address(class="form-control",
rows="3", placeholder="Юридический адрес") }}

                    </div>



                    <div class="row mb-3">

                        <div class="col-md-6">
```

```
                                <div class="mb-3">
                                    {{
form.email.label(class="form-label") }}
                                    {{ form.email(class="form-control",
placeholder="school@example.com") }}
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.website.label(class="form-label") }}
                                    {{
form.website(class="form-control", placeholder="http://example.com") }}
                                </div>
                            </div>
                        </div>

                        <!-- Дополнительная информация -->
                        <h5 class="mb-3 border-bottom pb-2
mt-4">Дополнительная информация</h5>
                        <div class="row">
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.founding_date.label(class="form-label") }}
                                    {{
form.founding_date(class="form-control") }}
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.number_of_students.label(class="form-label") }}
                                    {{
form.number_of_students(class="form-control", placeholder="500") }}
                                </div>
```

```
                            </div>
                        </div>

                        <!-- Тип школы и населенный пункт -->
                        <div class="row mb-3">
                            <div class="col-md-6">
                                {{
form.type_of_school.label(class="form-label") }}
                                {{
form.type_of_school(class="form-select") }}
                            </div>
                            <div class="col-md-6">
                                {{
form.settlement.label(class="form-label") }}
                                {{ form.settlement(class="form-select")
}}
                            </div>
                        </div>

                        <!-- Инфраструктура и специализации -->
                        <div class="row mt-3">
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.infrastructure.label(class="form-label") }}
                                    {{
form.infrastructure(class="form-select", multiple=True, size="5") }}
                                    <small
class="text-muted">Удерживайте Ctrl для выбора нескольких
вариантов</small>
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.specializations.label(class="form-label") }}
```

```
                                    {{
form.specializations(class="form-select", multiple=True, size="5") }}
                                </div>
                            </div>
                        </div>

                        <!-- Документы -->
                        <h5 class="mb-3 border-bottom pb-2
mt-4">Документы</h5>
                        <div class="row">
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.license.label(class="form-label") }}
                                    {{
form.license(class="form-control", rows="3", placeholder="Информация о
лицензии") }}
                                </div>
                            </div>
                            <div class="col-md-6">
                                <div class="mb-3">
                                    {{
form.accreditation.label(class="form-label") }}
                                    {{
form.accreditation(class="form-control", rows="3",
placeholder="Информация об аккредитации") }}
                                </div>
                            </div>
                        </div>

                        <div class="mt-4 d-flex
justify-content-between">
                            <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}" class="btn btn-secondary">
                                <i class="bi bi-eye"></i> Просмотр
                            </a>
```

```
                            <div>
                                <a href="{{ url_for('index') }}"
class="btn btn-outline-secondary me-2">
                                    <i class="bi bi-x-circle"></i>
Отмена
                                </a>  <!-- Закрывающий тег ПЕРЕД
содержимым! -->
                                {{ form.submit(class="btn btn-warning")
}}
                            </div>
                        </div>
                    </form>
                </div>
            </div>

            <!-- История изменений -->
            <div class="card mt-4">
                <div class="card-header bg-secondary text-white">
                    <h6 class="mb-0"><i class="bi
bi-clock-history"></i> История изменений</h6>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col-md-6">
                            <p><strong>Создана:</strong> {{
school.created_at.strftime('%d.%m.%Y %H:%M') if school.created_at else
'Нет данных' }}</p>
                            <p><strong>Создал:</strong> {{
school.creator_user.username if school.creator_user else 'Неизвестно'
}}</p>
                        </div>
                        <div class="col-md-6">
                            <p><strong>Обновлена:</strong> {{
school.updated_at.strftime('%d.%m.%Y %H:%M') if school.updated_at else
'Нет данных' }}</p>
                            <p><strong>Статус:</strong>
```

```
                                {% if school.is_active %}
                                <span class="badge
bg-success">Активна</span>
                                {% else %}
                                <span class="badge
bg-danger">Неактивна</span>
                                {% endif %}
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
document.addEventListener('DOMContentLoaded', function() {
    // Валидация форм
    var forms = document.querySelectorAll('.needs-validation');
    Array.prototype.slice.call(forms).forEach(function(form) {
        form.addEventListener('submit', function(event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        });
    });

    // Форматирование даты для поля
    var dateField = document.getElementById('founding_date');
    if (dateField && dateField.value) {
```

```
        // Преобразуем дату в формат YYYY-MM-DD для поля
        var date = new Date(dateField.value);
        dateField.value = date.toISOString().split('T')[0];
    }
});
</script>
{% endblock %}
```

templates/admin/inspections.html

```
{% extends "base.html" %}

{% block title %}Управление проверками - Реестр школ Алтайского края{% endblock %}

{% block content %}
<div class="container-fluid">
    <h1 class="h2 mb-4">Управление проверками Рособрнадзора</h1>

    <div class="d-flex justify-content-between align-items-center mb-4">
        <div class="btn-group" role="group">
            <a href="{{ url_for('admin_inspections') }}" class="btn btn-outline-secondary {% if status == 'all' %}active{% endif %}">
                Все проверки
            </a>
            <a href="{{ url_for('admin_inspections', status='with_violations') }}" class="btn btn-outline-danger {% if status == 'with_violations' %}active{% endif %}">
                С нарушениями
            </a>
            <a href="{{ url_for('admin_inspections', status='active') }}" class="btn btn-outline-warning {% if status == 'active' %}active{% endif %}">
                Не устранены
            </a>
```

```
            <a href="{{ url_for('admin_inspections', status='resolved')
}}" class="btn btn-outline-success {% if status == 'resolved'
%}active{% endif %}">
                Устранены
            </a>
        </div>

        <a href="{{ url_for('add_inspection') }}" class="btn
btn-primary">
            <i class="bi bi-plus-circle"></i> Добавить проверку
        </a>
    </div>

    <div class="card">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-clipboard-check"></i>
Список проверок</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Дата</th>
                            <th>Номер предписания</th>
                            <th>Школа</th>
                            <th>Нарушения</th>
                            <th>Статус</th>
                            <th>Действия</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for inspection in inspections.items %}
```

```
                        <tr>
                            <td>{{ loop.index + (inspections.page - 1)
* inspections.per_page }}</td>
                            <td>{{ inspection.Date.strftime('%d.%m.%Y')
}}</td>
                            <td><strong>{{
inspection.Prescription_Number }}</strong></td>
                            <td>
                                <a href="{{ url_for('school_detail',
school_id=inspection.PK_School) }}">
                                    {{ inspection.school.Official_Name
}}
                                </a>
                            </td>
                            <td>
                                {% if inspection.has_violations %}
                                    <span class="badge
bg-danger">Есть</span>
                                    {% if inspection.violation_type %}
                                        <br><small>{{
inspection.violation_type[:30] }}...</small>
                                    {% endif %}
                                {% else %}
                                    <span class="badge
bg-success">Нет</span>
                                {% endif %}
                            </td>
                            <td>
                                {% if inspection.is_resolved %}
                                    <span class="badge
bg-success">Устранено</span>
                                    {% if inspection.resolution_date %}
                                        <br><small>{{
inspection.resolution_date.strftime('%d.%m.%Y') }}</small>
                                    {% endif %}
                                {% else %}
```

```
                                    <span class="badge bg-warning
text-dark">Не устранено</span>
                                {% endif %}
                            </td>
                            <td>
                                <div class="btn-group btn-group-sm"
role="group">
                                    <a href="{{
url_for('edit_inspection', inspection_id=inspection.PK_Inspection) }}"
                                       class="btn btn-outline-warning">
                                        <i class="bi bi-pencil"></i>
                                    </a>
                                    <button type="button" class="btn
btn-outline-info"
                                            data-bs-toggle="popover"
                                            data-bs-html="true"

data-bs-content="<strong>Результат:</strong><br>{{
inspection.Result[:200] }}...<br><br>

<strong>Описание:</strong><br>{{ inspection.description[:200] if
inspection.description else 'Нет описания' }}">
                                        <i class="bi bi-eye"></i>
                                    </button>
                                    {% if
current_user.has_role('super_admin') %}
                                    <form method="POST" action="{{
url_for('delete_inspection', inspection_id=inspection.PK_Inspection)
}}"
                                          class="d-inline"
                                          onsubmit="return confirm('Вы
уверены, что хотите удалить эту проверку?')">
                                        <button type="submit"
class="btn btn-outline-danger">
                                            <i class="bi bi-trash"></i>
                                        </button>
                                    </form>
```

```
                                {% endif %}

                            </div>

                        </td>

                    </tr>

                    {% endfor %}

                </tbody>

            </table>

        </div>


        {% if inspections.pages > 1 %}

        <nav aria-label="Навигация по страницам" class="mt-4">

            <ul class="pagination justify-content-center">

                {% if inspections.has_prev %}

                <li class="page-item">

                    <a class="page-link" href="{{
url_for('admin_inspections', page=inspections.prev_num, status=status)
}}">

                        <i class="bi bi-chevron-left"></i>

                    </a>

                </li>

                {% endif %}


                {% for page_num in
inspections.iter_pages(left_edge=1, left_current=2, right_current=2,
right_edge=1) %}

                {% if page_num %}

                <li class="page-item {% if page_num ==
inspections.page %}active{% endif %}">

                    <a class="page-link" href="{{
url_for('admin_inspections', page=page_num, status=status) }}">

                        {{ page_num }}

                    </a>

                </li>

                {% else %}

                <li class="page-item disabled">
```

```
                        <span class="page-link">...</span>
                    </li>
                    {% endif %}
                    {% endfor %}

                    {% if inspections.has_next %}
                    <li class="page-item">
                        <a class="page-link" href="{{
url_for('admin_inspections', page=inspections.next_num, status=status)
}}">
                            <i class="bi bi-chevron-right"></i>
                        </a>
                    </li>
                    {% endif %}
                </ul>
            </nav>
            {% endif %}
        </div>
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        // Инициализация всплывающих окон
        var popoverTriggerList =
[].slice.call(document.querySelectorAll('[data-bs-toggle="popover"]'))
        var popoverList = popoverTriggerList.map(function
(popoverTriggerEl) {
            return new bootstrap.Popover(popoverTriggerEl)
        });
    });
</script>
{% endblock %}
```

templates/admin/school_history.hrml

```
{% extends "base.html" %}

{% block title %}История изменений - {{ school.Official_Name }}{% endblock %}

{% block content %}

<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-4">
        <h1>
            <i class="bi bi-clock-history"></i> История изменений
        </h1>
        <div>
            <a href="{{ url_for('edit_school', school_id=school.PK_School) }}"
               class="btn btn-warning">
                <i class="bi bi-pencil"></i> Редактировать
            </a>
            <a href="{{ url_for('school_detail', school_id=school.PK_School) }}"
               class="btn btn-outline-secondary">
                <i class="bi bi-arrow-left"></i> Назад к школе
            </a>
        </div>
    </div>

    <div class="card mb-4">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0">
                <i class="bi bi-building"></i> {{ school.Official_Name }}
            </h5>
        </div>
```

```
        <div class="card-body">
            <div class="row">
                <div class="col-md-6">
                    <p><strong>ID школы:</strong> {{ school.PK_School }}</p>
                    <p><strong>Адрес:</strong> {{ school.Legal_Adress }}</p>
                    <p><strong>Телефон:</strong> {{ school.Phone }}</p>
                </div>
                <div class="col-md-6">
                    <p><strong>Тип школы:</strong> {{ school.type_of_school.Name if school.type_of_school else 'Не указан' }}</p>
                    <p><strong>Населенный пункт:</strong>
                        {{ school.settlement.Type }} {{ school.settlement.Name }}
                    </p>
                    <p><strong>Статус:</strong>
                        {% if school.is_active %}
                        <span class="badge bg-success">Активна</span>
                        {% else %}
                        <span class="badge bg-danger">Неактивна</span>
                        {% endif %}
                    </p>
                </div>
            </div>
        </div>
    </div>

    <!-- История изменений -->
{% if versions %}
<div class="card">
    <div class="card-header bg-info text-white">
        <h5 class="mb-0">
```

```
            <i class="bi bi-list-check"></i> История изменений ({{
versions|length }} записей)
        </h5>
    </div>
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-hover">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Дата/Время</th>
                        <th>Действие</th>
                        <th>Пользователь</th>
                        <th>Изменения</th>
                        <th>Действия</th>
                    </tr>
                </thead>
                <tbody>
                    {% for version in versions %}
                    <tr>
                        <td>{{ loop.index }}</td>
                        <td>
                            <small>
                                {% if version[4] %}
                                    {{ version[4].strftime('%d.%m.%Y
%H:%M:%S') }}
                                {% else %}
                                    <span class="text-muted">Нет
даты</span>
                                {% endif %}
                            </small>
                        </td>
                        <td>
```

```
                            {% if version[6] == 'create' %}
                                <span class="badge
bg-success">Создание</span>
                            {% elif version[6] == 'update' %}
                                <span class="badge bg-warning
text-dark">Изменение</span>
                            {% elif version[6] == 'delete' %}
                                <span class="badge
bg-danger">Удаление</span>
                            {% elif version[6] == 'rollback' %}
                                <span class="badge
bg-secondary">Откат</span>
                            {% else %}
                                <span class="badge bg-info">{{
version[6] }}</span>
                            {% endif %}
                        </td>
                        <td>
                            {% if version[7] %}
                                {{ version[7] }}
                            {% elif version[5] %}
                                ID: {{ version[5] }}
                            {% else %}
                                <span class="text-muted">Система</span>
                            {% endif %}
                        </td>
                        <td>
                            <button type="button" class="btn btn-sm
btn-outline-info"
                                data-bs-toggle="modal"
                                data-bs-target="#versionModal{{
loop.index }}">
                                <i class="bi bi-eye"></i> Показать
изменения
                            </button>
```

```
                        </td>
                        <td>
                            {% if version[6] == 'update' and
current_user.has_role('super_admin') and loop.index > 1 %}
                            <form method="POST"
                                action="{{ url_for('rollback_version',
version_id=version[0]) }}?school_id={{ school.PK_School }}"
                                class="d-inline"
                                onsubmit="return confirm('Вы уверены,
что хотите откатиться к этой версии?')">
                                <button type="submit" class="btn btn-sm
btn-outline-danger">
                                    <i class="bi
bi-arrow-counterclockwise"></i> Откат
                                </button>
                            </form>
                            {% endif %}
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>
{% else %}
<div class="alert alert-info">
    <i class="bi bi-info-circle"></i> История изменений для этой школы
пока пуста.
</div>
{% endif %}
</div>

<!-- Модальные окна для просмотра изменений -->
{% for version in versions %}
```

```
<div class="modal fade" id="versionModal{{ loop.index }}" tabindex="-1"
    aria-labelledby="versionModalLabel{{ loop.index }}"
aria-hidden="true">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="versionModalLabel{{
loop.index }}">
                    {% if version[6] == 'create' %}
                    Создание
                    {% elif version[6] == 'update' %}
                    Изменение
                    {% elif version[6] == 'delete' %}
                    Удаление
                    {% elif version[6] == 'rollback' %}
                    Откат
                    {% else %}
                    {{ version[6] }}
                    {% endif %}
                    от {{ version[4].strftime('%d.%m.%Y %H:%M') if
version[4] else '' }}
                </h5>
                <button type="button" class="btn-close"
data-bs-dismiss="modal" aria-label="Закрыть"></button>
            </div>
            <div class="modal-body">
                {% set before_data = version[2]|fromjson if version[2]
else {} %}
                {% set after_data = version[3]|fromjson if version[3]
else {} %}

                <!-- Улучшенное отображение измененных полей -->
                {% if version[6] == 'update' and before_data and
after_data %}
                <h6>Измененные поля:</h6>
```

```
                <div class="table-responsive">
                    <table class="table table-sm table-bordered">
                        <thead>
                            <tr>
                                <th>Поле</th>
                                <th>Было</th>
                                <th>Стало</th>
                                <th>Тип изменения</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% set changed_count = 0 %}

                            <!-- Собираем все уникальные ключи -->
                            {% set all_keys = [] %}
                            {% for key in before_data %}
                                {% if key not in all_keys %}
                                    {% set _ = all_keys.append(key) %}
                                {% endif %}
                            {% endfor %}
                            {% for key in after_data %}
                                {% if key not in all_keys %}
                                    {% set _ = all_keys.append(key) %}
                                {% endif %}
                            {% endfor %}

                            <!-- Показываем только действительно
измененные поля -->
                            {% for key in all_keys %}
                                {% set before_value =
before_data.get(key) %}
                                {% set after_value =
after_data.get(key) %}
```

```
                                {% set is_changed = false %}
                                {% set change_type = '' %}

                                <!-- Определяем тип изменения -->
                                {% if before_value != after_value %}
                                    {% if before_value in [None, '']
and after_value not in [None, ''] %}
                                        {% set change_type = 'added' %}
                                        {% set is_changed = true %}
                                    {% elif before_value not in [None,
''] and after_value in [None, ''] %}
                                        {% set change_type = 'removed'
%}
                                        {% set is_changed = true %}
                                    {% elif before_value != after_value
%}
                                        {% set change_type = 'modified'
%}
                                        {% set is_changed = true %}
                                    {% endif %}
                                {% endif %}

                                {% if is_changed %}
                                    {% set changed_count =
changed_count + 1 %}
                                    <tr>
                                        <td><strong>{{
get_field_display_name(key) }}</strong></td>
                                        <td>
                                            {% if before_value is none
or before_value == '' %}
                                                <span
class="text-muted"><i>(пусто)</i></span>
                                            {% elif key in
['PK_Type_of_School', 'PK_Settlement'] %}
```

```
                                        <!-- Преобразуем ID в
названия -->
                                        {% if key ==
'PK_Type_of_School' %}
                                            {% set type_obj =
get_school_type_by_id(before_value) %}
                                            {{ type_obj.Name if
type_obj else before_value }}
                                        {% elif key ==
'PK_Settlement' %}
                                            {% set
settlement_obj = get_settlement_by_id(before_value) %}
                                            {{
settlement_obj.Name if settlement_obj else before_value }}
                                        {% endif %}
                                    {% elif key == 'is_active'
%}
                                        {% if before_value %}
                                            <span class="badge
bg-success">Активна</span>
                                        {% else %}
                                            <span class="badge
bg-danger">Неактивна</span>
                                        {% endif %}
                                    {% else %}
                                        {{ before_value }}
                                    {% endif %}
                                </td>
                                <td>
                                    {% if after_value is none
or after_value == '' %}
                                        <span
class="text-muted"><i>(пусто)</i></span>
                                    {% elif key in
['PK_Type_of_School', 'PK_Settlement'] %}
                                        <!-- Преобразуем ID в
названия -->
```

```
                                                {% if key ==
'PK_Type_of_School' %}
                                                    {% set type_obj =
get_school_type_by_id(after_value) %}
                                                    {{ type_obj.Name if
type_obj else after_value }}
                                                {% elif key ==
'PK_Settlement' %}
                                                    {% set
settlement_obj = get_settlement_by_id(after_value) %}
                                                    {{
settlement_obj.Name if settlement_obj else after_value }}
                                                {% endif %}
                                            {% elif key == 'is_active'
%}
                                                {% if after_value %}
                                                    <span class="badge
bg-success">Активна</span>
                                                {% else %}
                                                    <span class="badge
bg-danger">Неактивна</span>
                                                {% endif %}
                                            {% else %}
                                                {{ after_value }}
                                            {% endif %}
                                        </td>
                                        <td>
                                            {% if change_type ==
'added' %}
                                                <span class="badge
bg-success">Добавлено</span>
                                            {% elif change_type ==
'removed' %}
                                                <span class="badge
bg-danger">Удалено</span>
                                            {% elif change_type ==
'modified' %}
```

```
                                        <span class="badge
bg-warning">Изменено</span>
                                    {% endif %}
                                </td>
                            </tr>
                        {% endif %}
                    {% endfor %}

                    {% if changed_count == 0 %}
                    <tr>
                        <td colspan="4" class="text-center
text-muted">Изменений не обнаружено</td>
                    </tr>
                    {% endif %}
                </tbody>
            </table>
        </div>
        {% elif version[6] == 'create' %}
        <h6>Созданная запись:</h6>
        {% if after_data %}
        <div class="table-responsive">
            <table class="table table-sm table-bordered">
                <thead>
                    <tr>
                        <th>Поле</th>
                        <th>Значение</th>
                    </tr>
                </thead>
                <tbody>
                    {% for key, value in after_data.items() %}
                    <tr>
                        <td><strong>{{
get_field_display_name(key) }}</strong></td>
```

```
                                <td>
                                    {% if value is none or value == ''
%}
                                        <span
class="text-muted"><i>(пусто)</i></span>
                                    {% elif key in
['PK_Type_of_School', 'PK_Settlement'] %}
                                        {% if key ==
'PK_Type_of_School' %}
                                            {% set type_obj =
get_school_type_by_id(value) %}
                                            {{ type_obj.Name if
type_obj else value }}
                                        {% elif key == 'PK_Settlement'
%}
                                            {% set settlement_obj =
get_settlement_by_id(value) %}
                                            {{ settlement_obj.Name if
settlement_obj else value }}
                                        {% endif %}
                                    {% elif key == 'is_active' %}
                                        {% if value %}
                                            <span class="badge
bg-success">Активна</span>
                                        {% else %}
                                            <span class="badge
bg-danger">Неактивна</span>
                                        {% endif %}
                                    {% else %}
                                        {{ value }}
                                    {% endif %}
                                </td>
                            </tr>
                            {% endfor %}
                        </tbody>
                    </table>
```

```
                </div>
                {% else %}
                <p class="text-muted">Нет данных</p>
                {% endif %}
                {% elif version[6] == 'delete' %}
                <div class="alert alert-warning">
                    <i class="bi bi-exclamation-triangle"></i> Запись
была удалена.
                </div>
                {% if before_data %}
                <h6>Удаленные данные:</h6>
                <div class="table-responsive">
                    <table class="table table-sm table-bordered">
                        <thead>
                            <tr>
                                <th>Поле</th>
                                <th>Значение</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for key, value in before_data.items() %}
                            <tr>
                                <td><strong>{{
get_field_display_name(key) }}</strong></td>
                                <td>
                                    {% if value is none or value == ''
%}
                                        <span
class="text-muted"><i>(пусто)</i></span>
                                    {% else %}
                                        {{ value }}
                                    {% endif %}
                                </td>
                            </tr>
```

```
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
                {% endif %}
                {% elif version[6] == 'rollback' %}
                <div class="alert alert-info">
                    <i class="bi bi-info-circle"></i> Выполнен откат к
предыдущей версии.
                </div>
                {% if before_data and after_data %}
                <h6>Восстановленные поля:</h6>
                <div class="table-responsive">
                    <table class="table table-sm table-bordered">
                        <thead>
                            <tr>
                                <th>Поле</th>
                                <th>Было</th>
                                <th>Стало</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for key in before_data %}
                                {% if key in after_data %}
                                    {% set before_value =
before_data[key] %}
                                    {% set after_value =
after_data[key] %}
                                    {% if before_value != after_value
%}
                                    <tr>
                                        <td><strong>{{
get_field_display_name(key) }}</strong></td>
                                        <td>
```

```
                                        {% if before_value is none
or before_value == '' %}
                                            <span
class="text-muted"><i>(пусто)</i></span>
                                        {% else %}
                                            {{ before_value }}
                                        {% endif %}
                                    </td>
                                    <td>
                                        {% if after_value is none
or after_value == '' %}
                                            <span
class="text-muted"><i>(пусто)</i></span>
                                        {% else %}
                                            {{ after_value }}
                                        {% endif %}
                                    </td>
                                </tr>
                                {% endif %}
                            {% endif %}
                        {% endfor %}
                    </tbody>
                </table>
            </div>
            {% endif %}
            {% else %}
            <p class="text-muted">Нет данных для отображения</p>
            {% endif %}

            <!-- Детальная информация -->
            <div class="row mt-4">
                <div class="col-md-6">
                    <div class="card">
                        <div class="card-header bg-light">
```

```
                                <h6 class="mb-0">Детальная
информация</h6>

                            </div>

                            <div class="card-body">

                                <p><strong>Действие:</strong> {{
version[6] }}</p>

                                <p><strong>Дата:</strong> {{
version[4].strftime('%d.%m.%Y %H:%M:%S') if version[4] else 'Нет даты'
}}</p>

                                <p><strong>Пользователь:</strong>

                                    {% if version[5] %}

                                        {% set user =
get_user_by_id(version[5]) %}

                                        {{ user.username if user else
'ID: ' ~ version[5] }}

                                    {% else %}

                                        Система

                                    {% endif %}

                                </p>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

            <div class="modal-footer">

                <button type="button" class="btn btn-secondary"
data-bs-dismiss="modal">Закрыть</button>

            </div>

        </div>

    </div>

</div>

{% endfor %}



<style>

.table-hover tbody tr {
```

```
    transition: background-color 0.2s ease;
}
.modal {
    backdrop-filter: blur(3px);
}
.modal-content {
    animation: modalFadeIn 0.3s ease;
}
@keyframes modalFadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
.table-bordered td, .table-bordered th {
    vertical-align: middle;
}
.table-bordered td {
    word-break: break-word;
}
</style>

<script>
document.addEventListener('DOMContentLoaded', function() {
    var modals = document.querySelectorAll('.modal');
    modals.forEach(function(modal) {
        new bootstrap.Modal(modal);
    });
```

```
    var showButtons =
document.querySelectorAll('[data-bs-toggle="modal"]');

    showButtons.forEach(function(button) {

        button.addEventListener('click', function(e) {

            e.stopPropagation();

        });

    });

});

</script>


{% endblock %}
```

templates/auth/login.html

```
{% extends "base.html" %}


{% block title %}Вход - Реестр школ Алтайского края{% endblock %}


{% block content %}

<div class="row justify-content-center">

    <div class="col-md-6 col-lg-5">

        <div class="card">

            <div class="card-header bg-primary text-white text-center">

                <h4 class="mb-0"><i class="bi
bi-box-arrow-in-right"></i> Вход в систему</h4>

            </div>

            <div class="card-body">

                <form method="POST" action="{{ url_for('login') }}"
class="needs-validation" novalidate>

                    {{ form.hidden_tag() }}


                    <div class="mb-3">

                        <label for="username" class="form-label">Имя
пользователя</label>

                        <div class="input-group">
```

```
                            <span class="input-group-text">
                                <i class="bi bi-person"></i>
                            </span>
                            {{ form.username(class="form-control",
placeholder="Введите имя пользователя", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пожалуйста, введите имя пользователя
                        </div>
                    </div>

                    <div class="mb-3">
                        <label for="password"
class="form-label">Пароль</label>
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-key"></i>
                            </span>
                            {{ form.password(class="form-control",
placeholder="Введите пароль", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пожалуйста, введите пароль
                        </div>
                    </div>

                    <div class="mb-3 form-check">
                        {{ form.remember_me(class="form-check-input")
}}
                        <label class="form-check-label"
for="remember_me">
                            Запомнить меня
                        </label>
                    </div>
```

```
                    <div class="d-grid gap-2">
                        {{ form.submit(class="btn btn-primary btn-lg")
}}
                    </div>
                </form>

                <hr class="my-4">

                <div class="text-center">
                    <p class="mb-2">Нет аккаунта?</p>
                    <a href="{{ url_for('register') }}" class="btn
btn-outline-primary">
                        <i class="bi bi-person-plus"></i>
Зарегистрироваться
                    </a>

                    <div class="mt-3">
                        <a href="{{ url_for('auth_gosuslugi') }}"
class="btn btn-outline-success">
                            <i class="bi bi-passport"></i> Войти через
госуслуги
                        </a>
                    </div>
                </div>
            </div>

            <div class="card-footer text-center">
                <small class="text-muted">
                    По всем вопросам обращайтесь по телефону 8 (3852)
29-44-07
                </small>
            </div>
        </div>
```

```
        <!-- Демо доступы -->
        <div class="card mt-4">
            <div class="card-header bg-info text-white">
                <h6 class="mb-0"><i class="bi bi-info-circle"></i> Демо доступы</h6>
            </div>
            <div class="card-body">
                <div class="row g-2">
                    <div class="col-12">
                        <div class="card bg-light">
                            <div class="card-body py-2">
                                <small>
                                    <strong>Администратор:</strong> admin / admin123<br>
                                    <strong>Родитель:</strong> parent / parent123<br>
                                    <strong>Учитель:</strong> teacher / teacher123
                                </small>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

templates/auth/ptofile.html

```
{% extends "base.html" %}

{% block title %}Профиль - Реестр школ Алтайского края{% endblock %}
```

```
{% block content %}

<div class="row">

    <div class="col-lg-8">

        <div class="card">

            <div class="card-header bg-primary text-white">

                <h5 class="mb-0"><i class="bi bi-person-circle"></i>
Профиль пользователя</h5>

            </div>

            <div class="card-body">

                <form method="POST" action="{{
url_for('update_profile') }}" class="needs-validation" novalidate>

                    {{ form.hidden_tag() }}

                    <div class="row">

                        <div class="col-md-6 mb-3">

                            <label for="username"
class="form-label">Имя пользователя</label>

                            {{ form.username(class="form-control",
required=true) }}

                        </div>

                        <div class="col-md-6 mb-3">

                            <label for="email"
class="form-label">Email</label>

                            {{ form.email(class="form-control",
required=true) }}

                        </div>

                    </div>

                    <hr>

                    <h6>Смена пароля</h6>

                    <div class="row">

                        <div class="col-md-6 mb-3">

                            <label for="current_password"
class="form-label">Текущий пароль</label>

                            {{
form.current_password(class="form-control") }}

                            <small class="text-muted">Оставьте пустым,
если не хотите менять пароль.</small>
```

```
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-md-6 mb-3">
                            <label for="new_password"
class="form-label">Новый пароль</label>
                            {{ form.new_password(class="form-control")
}}
                        </div>
                        <div class="col-md-6 mb-3">
                            <label for="new_password2"
class="form-label">Повторите новый пароль</label>
                            {{ form.new_password2(class="form-control")
}}
                        </div>
                    </div>
                    <button type="submit" class="btn
btn-primary">Обновить профиль</button>
                </form>
            </div>
        </div>
    </div>
    <div class="col-lg-4">
        <div class="card">
            <div class="card-header bg-info text-white">
                <h5 class="mb-0"><i class="bi bi-info-circle"></i>
Статистика</h5>
            </div>
            <div class="card-body">
                <p>Вы зарегистрированы: {{
current_user.created_at.strftime('%d.%m.%Y') }}</p>
                <p>Последний вход: {{
current_user.last_login.strftime('%d.%m.%Y %H:%M') if
current_user.last_login else 'Никогда' }}</p>
                <p>Количество оставленных отзывов: {{ user_reviews
}}</p>
```

```
            </div>

        </div>

    </div>

</div>

{% endblock %}
```

templates/auth/register.html

```
{% extends "base.html" %}



{% block title %}Регистрация - Реестр школ Алтайского края{% endblock
%}



{% block content %}

<div class="row justify-content-center">

    <div class="col-md-8 col-lg-6">

        <div class="card">

            <div class="card-header bg-success text-white text-center">

                <h4 class="mb-0"><i class="bi bi-person-plus"></i>
Регистрация</h4>

            </div>

            <div class="card-body">

                <!-- Flash сообщения -->

                {% with messages =
get_flashed_messages(with_categories=true) %}

                    {% if messages %}

                        {% for category, message in messages %}

                            <div class="alert alert-{{ 'danger' if
category == 'error' else category }} alert-dismissible fade show mb-4"
role="alert">

                                {{ message }}

                                <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>

                            </div>

                        {% endfor %}

                    {% endif %}
```

```
                {% endwith %}


                <form method="POST" action="{{ url_for('register') }}"
class="needs-validation" novalidate>
                    {{ form.hidden_tag() }}


                    <!-- Имя пользователя -->
                    <div class="mb-3">
                        <label for="username" class="form-label">Имя
пользователя</label>
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-person"></i>
                            </span>
                            {{ form.username(class="form-control",
placeholder="Введите имя пользователя", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пожалуйста, введите имя пользователя
(минимум 3 символа).
                        </div>
                        {% if form.username.errors %}
                        <div class="text-danger small mt-1">
                            {% for error in form.username.errors %}
                                {{ error }}
                            {% endfor %}
                        </div>
                        {% endif %}
                    </div>


                    <!-- Email -->
                    <div class="mb-3">
                        <label for="email"
class="form-label">Email</label>
```

```
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-envelope"></i>
                            </span>
                            {{ form.email(class="form-control",
placeholder="Введите email", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пожалуйста, введите корректный email.
                        </div>
                        {% if form.email.errors %}
                        <div class="text-danger small mt-1">
                            {% for error in form.email.errors %}
                                {{ error }}
                            {% endfor %}
                        </div>
                        {% endif %}
                    </div>

                    <!-- Пароль -->
                    <div class="mb-3">
                        <label for="password"
class="form-label">Пароль</label>
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-key"></i>
                            </span>
                            {{ form.password(class="form-control",
placeholder="Введите пароль (минимум 6 символов)", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пароль должен быть не менее 6 символов.
                        </div>
```

```
                        <div class="form-text">
                            <div id="password-strength" class="mt-2">
                                <div class="progress" style="height:
8px;">
                                    <div id="password-strength-bar"
class="progress-bar" role="progressbar" style="width: 0%"></div>
                                </div>
                                <small id="password-strength-text"
class="text-muted">Сложность пароля: очень слабый</small>
                            </div>
                        </div>
                        {% if form.password.errors %}
                        <div class="text-danger small mt-1">
                            {% for error in form.password.errors %}
                                {{ error }}
                            {% endfor %}
                        </div>
                        {% endif %}
                    </div>

                    <!-- Повтор пароля -->
                    <div class="mb-3">
                        <label for="password2"
class="form-label">Повторите пароль</label>
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-key-fill"></i>
                            </span>
                            {{ form.password2(class="form-control",
placeholder="Повторите пароль", required=true) }}
                        </div>
                        <div class="invalid-feedback">
                            Пароли должны совпадать.
                        </div>
```

```
                    <div id="password-match" class="form-text">
                        <small id="password-match-text"></small>
                    </div>
                    {% if form.password2.errors %}
                    <div class="text-danger small mt-1">
                        {% for error in form.password2.errors %}
                            {{ error }}
                        {% endfor %}
                    </div>
                    {% endif %}
                </div>

                <!-- Роль -->
                <div class="mb-3">
                    <label for="role" class="form-label">Выберите
вашу роль</label>
                    {{ form.role(class="form-select",
required=true) }}
                    <div class="form-text">
                        <small>
                            <strong>Ученик/Родитель:</strong>
только просмотр информации<br>
                            <strong>Работник учреждения:</strong>
может редактировать данные своей школы<br>
                            <strong>Другое:</strong> только
просмотр информации
                        </small>
                    </div>
                </div>

                <!-- Выбор учреждения (только для работников) -->
                <div class="mb-3" id="school-field" style="display:
none;">
                    <label for="school_id"
class="form-label">Выберите ваше образовательное учреждение</label>
```

```
                        {{ form.school_id(class="form-select",
required=false) }}
                        <div class="invalid-feedback">
                            Для работника учреждения необходимо выбрать
учреждение.
                        </div>
                        {% if form.school_id.errors %}
                        <div class="text-danger small mt-1">
                            {% for error in form.school_id.errors %}
                                {{ error }}
                            {% endfor %}
                        </div>
                        {% endif %}
                    </div>

                    <div class="d-grid gap-2">
                        {{ form.submit(class="btn btn-success btn-lg")
}}
                    </div>
                </form>

                <hr class="my-4">

                <div class="text-center">
                    <p class="mb-2">Уже есть аккаунт?</p>
                    <a href="{{ url_for('login') }}" class="btn
btn-outline-primary">
                        <i class="bi bi-box-arrow-in-right"></i> Войти
                    </a>
                </div>
            </div>
            <div class="card-footer text-center">
                <small class="text-muted">
```

```
                    Регистрируясь, вы соглашаетесь с условиями
использования и политикой конфиденциальности.

                </small>

            </div>

        </div>


        <!-- Информация о ролях -->

        <div class="card mt-4">

            <div class="card-header bg-info text-white">

                <h6 class="mb-0"><i class="bi bi-info-circle"></i>
Информация о ролях</h6>

            </div>

            <div class="card-body">

                <div class="row g-2">

                    <div class="col-md-6">

                        <div class="card bg-light">

                            <div class="card-body py-3">

                                <h6
class="card-title">Ученик/Родитель</h6>

                                <ul class="small mb-0">

                                    <li>Просмотр информации о
школах</li>

                                    <li>Оставление отзывов</li>

                                    <li>Поиск школ по фильтрам</li>

                                    <li>Не может редактировать
данные</li>

                                </ul>

                            </div>

                        </div>

                    </div>

                    <div class="col-md-6">

                        <div class="card bg-light">

                            <div class="card-body py-3">

                                <h6 class="card-title">Работник
учреждения</h6>
```

```
                                <ul class="small mb-0">
                                    <li>Все права ученика/родителя</li>
                                    <li>Редактирование данных своей
школы</li>
                                    <li>Управление отзывами своей
школы</li>
                                    <li>Добавление сотрудников и
программ</li>
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>


<script>
document.addEventListener('DOMContentLoaded', function() {
    const roleSelect = document.getElementById('role');
    const schoolField = document.getElementById('school-field');
    const schoolSelect = document.getElementById('school_id');
    const passwordInput = document.getElementById('password');
    const passwordStrengthBar =
document.getElementById('password-strength-bar');
    const passwordStrengthText =
document.getElementById('password-strength-text');
    const password2Input = document.getElementById('password2');
    const passwordMatchText =
document.getElementById('password-match-text');


    // Показать/скрыть поле выбора школы
    function toggleSchoolField() {
        if (roleSelect.value === '2') {
```

```
            schoolField.style.display = 'block';
            schoolSelect.required = true;
        } else {
            schoolField.style.display = 'none';
            schoolSelect.required = false;
        }
    }

    roleSelect.addEventListener('change', toggleSchoolField);
    toggleSchoolField(); // Инициализация

    // Проверка сложности пароля
    function checkPasswordStrength(password) {
        let strength = 0;

        if (password.length >= 8) strength++;
        if (password.match(/[a-z]+/)) strength++;
        if (password.match(/[A-Z]+/)) strength++;
        if (password.match(/[0-9]+/)) strength++;
        if (password.match(/[!@#$%^&*()_+\-=\[\]{};':"\\|,.<>\/?]+/))
strength++;

        return strength;
    }

    function updatePasswordStrength() {
        const password = passwordInput.value;
        const strength = checkPasswordStrength(password);
        const strengthPercent = (strength / 5) * 100;

        passwordStrengthBar.style.width = strengthPercent + '%';

        let color, text;
```

```
        switch(strength) {
            case 0:
            case 1:
                color = 'bg-danger';
                text = 'Очень слабый';
                break;
            case 2:
                color = 'bg-warning';
                text = 'Слабый';
                break;
            case 3:
                color = 'bg-info';
                text = 'Средний';
                break;
            case 4:
                color = 'bg-primary';
                text = 'Хороший';
                break;
            case 5:
                color = 'bg-success';
                text = 'Отличный';
                break;
        }

        passwordStrengthBar.className = 'progress-bar ' + color;
        passwordStrengthText.textContent = 'Сложность пароля: ' + text;
    }

    passwordInput.addEventListener('input', updatePasswordStrength);

    // Проверка совпадения паролей
    function checkPasswordMatch() {
```

```
        const password = passwordInput.value;
        const password2 = password2Input.value;

        if (!password2) {
            passwordMatchText.textContent = '';
            passwordMatchText.className = '';
            return;
        }

        if (password === password2) {
            passwordMatchText.textContent = '✓ Пароли совпадают';
            passwordMatchText.className = 'text-success';
        } else {
            passwordMatchText.textContent = '✗ Пароли не совпадают';
            passwordMatchText.className = 'text-danger';
        }
    }

    passwordInput.addEventListener('input', checkPasswordMatch);
    password2Input.addEventListener('input', checkPasswordMatch);

    // Валидация формы
    const forms = document.querySelectorAll('.needs-validation');
    Array.prototype.slice.call(forms).forEach(function(form) {
        form.addEventListener('submit', function(event) {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();

                // Показать сообщения об ошибках
                if (roleSelect.value === '2' && schoolSelect.value ===
'0') {
                    event.preventDefault();
```

```
                    schoolSelect.classList.add('is-invalid');
                    const errorDiv = schoolSelect.nextElementSibling;
                    if (errorDiv &&
errorDiv.classList.contains('invalid-feedback')) {
                        errorDiv.textContent = 'Для работника
учреждения необходимо выбрать учреждение.';
                        errorDiv.style.display = 'block';
                    }
                }
            }
            form.classList.add('was-validated');
        }, false);
    });
});
</script>
{% endblock %}
```

templates/errors/403.html

```
{% extends "base.html" %}

{% block title %}Доступ запрещен - Реестр школ Алтайского края{%
endblock %}

{% block content %}
<div class="container py-5">
    <div class="row justify-content-center">
        <div class="col-md-8 text-center">
            <div class="mb-4">
                <i class="bi bi-shield-slash display-1
text-danger"></i>
            </div>
            <h1 class="display-4 fw-bold text-danger mb-3">403</h1>
            <h2 class="h3 mb-4">Доступ запрещен</h2>
            <p class="lead mb-4">
```

```
                У вас недостаточно прав для доступа к этой странице.
            </p>
            <div class="d-flex justify-content-center gap-3">
                <a href="{{ url_for('index') }}" class="btn btn-primary
btn-lg">
                    <i class="bi bi-house"></i> На главную
                </a>
                <a href="javascript:history.back()" class="btn
btn-outline-secondary btn-lg">
                    <i class="bi bi-arrow-left"></i> Назад
                </a>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

templates/errors/404.html

```
{% extends "base.html" %}


{% block title %}Доступ запрещен - Реестр школ Алтайского края{%
endblock %}


{% block content %}
<div class="container py-5">
    <div class="row justify-content-center">
        <div class="col-md-8 text-center">
            <div class="mb-4">
                <i class="bi bi-shield-slash display-1
text-danger"></i>
            </div>
            <h1 class="display-4 fw-bold text-danger mb-3">403</h1>
            <h2 class="h3 mb-4">Доступ запрещен</h2>
            <p class="lead mb-4">
```

```
                У вас недостаточно прав для доступа к этой странице.
            </p>
            <div class="d-flex justify-content-center gap-3">
                <a href="{{ url_for('index') }}" class="btn btn-primary
btn-lg">
                    <i class="bi bi-house"></i> На главную
                </a>
                <a href="javascript:history.back()" class="btn
btn-outline-secondary btn-lg">
                    <i class="bi bi-arrow-left"></i> Назад
                </a>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

templates/errors/500.html

```
{% extends "base.html" %}

{% block title %}Ошибка сервера - Реестр школ Алтайского края{%
endblock %}

{% block content %}
<div class="container py-5">
    <div class="row justify-content-center">
        <div class="col-md-8 text-center">
            <div class="mb-4">
                <i class="bi bi-exclamation-triangle display-1
text-warning"></i>
            </div>
            <h1 class="display-4 fw-bold text-warning mb-3">500</h1>
            <h2 class="h3 mb-4">Внутренняя ошибка сервера</h2>
            <p class="lead mb-4">
```

```
                Произошла непредвиденная ошибка. Пожалуйста, попробуйте
позже.
            </p>
            <div class="d-flex justify-content-center gap-3">
                <a href="{{ url_for('index') }}" class="btn btn-primary
btn-lg">
                    <i class="bi bi-house"></i> На главную
                </a>
                <button onclick="location.reload()" class="btn
btn-outline-warning btn-lg">
                    <i class="bi bi-arrow-clockwise"></i> Обновить
страницу
                </button>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

templates/reports/boarding_schools.html

```
{% extends "base.html" %}

{% block title %}Школы-интернаты - Реестр школ Алтайского края{%
endblock %}

{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center
mb-4">
        <h1>Школы-интернаты с более чем 200 учащимися</h1>
        <a href="{{ url_for('export_schools') }}?format=excel"
class="btn btn-success">
            <i class="bi bi-download"></i> Экспорт в Excel
        </a>
    </div>
```

```
    <div class="card mb-4">
        <div class="card-header bg-info text-white">
            <h5 class="mb-0"><i class="bi bi-info-circle"></i>
Информация об отчете</h5>
        </div>
        <div class="card-body">
            <p>Отчет показывает школы-интернаты с количеством учащихся
более 200 человек.</p>
            <p>Тип школы: <strong>{{ school_type.Name }}</strong></p>
            <p>Найдено школ: <strong>{{ schools|length }}</strong></p>
        </div>
    </div>

    {% if schools %}
    <div class="card">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-table"></i> Список
школ</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Название школы</th>
                            <th>Адрес</th>
                            <th>Телефон</th>
                            <th>Учащихся</th>
                            <th>Населенный пункт</th>
                            <th>Действия</th>
                        </tr>
                    </thead>
```

```
                    <tbody>
                        {% for school in schools %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td><strong>{{ school.Official_Name
}}</strong></td>
                            <td>{{ school.Legal_Adress[:50] }}{% if
school.Legal_Adress|length > 50 %}...{% endif %}</td>
                            <td>{{ school.Phone }}</td>
                            <td>
                                <span class="badge bg-info">{{
school.Number_of_Students }}</span>
                            </td>
                            <td>
                                {% if school.settlement %}
                                {{ school.settlement.Type }} {{
school.settlement.Name }}
                                {% endif %}
                            </td>
                            <td>
                                <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}"
                                   class="btn btn-sm
btn-outline-primary">
                                    <i class="bi bi-eye"></i>
                                </a>
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    {% else %}
```

```
    <div class="alert alert-warning">
        <i class="bi bi-exclamation-triangle"></i>
        Не найдено школ-интернатов с более чем 200 учащимися.
    </div>
    {% endif %}

    <div class="mt-4">
        <a href="{{ url_for('index') }}" class="btn
btn-outline-secondary">
            <i class="bi bi-arrow-left"></i> Вернуться на главную
        </a>
    </div>
</div>
{% endblock %}
```

templates/reports/index.html

```
{% extends "base.html" %}

{% block title %}Генерация отчетов - Реестр школ Алтайского края{%
endblock %}

{% block content %}
<div class="container">
    <h1 class="mb-4"><i class="bi bi-file-text"></i> Генерация
отчетов</h1>

    <div class="card mb-4">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-info-circle"></i>
Информация</h5>
        </div>
        <div class="card-body">
            <p>В этом разделе вы можете сформировать различные отчеты
по образовательным учреждениям Алтайского края.</p>
```

```
            <p>Выберите тип отчета, задайте необходимые параметры и
нажмите "Сформировать отчет".</p>

        </div>

    </div>


    <div class="row">

        <!-- Статистика по количеству учащихся -->

        <div class="col-md-6 mb-4">

            <div class="card h-100">

                <div class="card-header bg-primary text-white">

                    <h5 class="mb-0"><i class="bi bi-people"></i>
Статистика по количеству учащихся</h5>

                </div>

                <div class="card-body">

                    <p>Анализ количества учащихся по районам, типам
школ и населенным пунктам.</p>

                    <ul class="mb-3">

                        <li>Общее количество учащихся</li>

                        <li>Среднее количество по районам</li>

                        <li>Распределение по типам школ</li>

                        <li>Динамика набора по годам</li>

                    </ul>

                    <a href="{{ url_for('report_students') }}"
class="btn btn-primary">

                        <i class="bi bi-bar-chart"></i> Сформировать
отчет

                    </a>

                </div>

            </div>

        </div>


        <!-- Статистика по оснащенности -->

        <div class="col-md-6 mb-4">

            <div class="card h-100">
```

```
                <div class="card-header bg-success text-white">
                    <h5 class="mb-0"><i class="bi bi-building"></i>
Оснащенность школ</h5>
                </div>
                <div class="card-body">
                    <p>Анализ уровня оснащенности школ
инфраструктурой.</p>
                    <ul class="mb-3">
                        <li>Наличие библиотек, спортзалов,
лабораторий</li>
                        <li>Процент школ с компьютерными классами</li>
                        <li>Сравнение по районам</li>
                        <li>Рекомендации по оснащению</li>
                    </ul>
                    <a href="{{ url_for('report_infrastructure') }}"
class="btn btn-success">
                        <i class="bi bi-pie-chart"></i> Сформировать
отчет
                    </a>
                </div>
            </div>
        </div>

        <!-- Кадровый состав -->
        <div class="col-md-6 mb-4">
            <div class="card h-100">
                <div class="card-header bg-info text-white">
                    <h5 class="mb-0"><i class="bi bi-person-badge"></i>
Кадровый состав</h5>
                </div>
                <div class="card-body">
                    <p>Отчет по сотрудникам образовательных
учреждений.</p>
                    <ul class="mb-3">
                        <li>Учителя, директора, завучи</li>
```

```
                        <li>Распределение по должностям</li>
                        <li>Статистика по стажу работы</li>
                        <li>Квалификационный состав</li>
                    </ul>
                    <a href="{{ url_for('report_staff') }}" class="btn
btn-info">
                        <i class="bi bi-person-lines-fill"></i>
Сформировать отчет
                    </a>
                </div>
            </div>
        </div>

        <!-- Программы образования -->
        <div class="col-md-6 mb-4">
            <div class="card h-100">
                <div class="card-header bg-warning text-dark">
                    <h5 class="mb-0"><i class="bi bi-book"></i>
Образовательные программы</h5>
                </div>
                <div class="card-body">
                    <p>Анализ образовательных программ по школам.</p>
                    <ul class="mb-3">
                        <li>Основные и дополнительные программы</li>
                        <li>Распределение по типам программ</li>
                        <li>Список программ по школам</li>
                        <li>Популярность программ</li>
                    </ul>
                    <a href="{{ url_for('report_programs') }}"
class="btn btn-warning">
                        <i class="bi bi-list-check"></i> Сформировать
отчет
                    </a>
                </div>
```

```
            </div>
        </div>
          <!-- ВСТАВЬТЕ ЗДЕСЬ КАРТОЧКУ ПРО НАРУШЕНИЯ -->
    <div class="col-md-6 mb-4">
        <div class="card h-100">
            <div class="card-header bg-danger text-white">
                <h5 class="mb-0"><i class="bi
bi-exclamation-triangle"></i> Нарушения Рособрнадзора</h5>
            </div>
            <div class="card-body">
                <p>Сводный отчет по нарушениям, выявленным в ходе
проверок.</p>
                <ul class="mb-3">
                    <li>Список нарушений по школам</li>
                    <li>Статистика по типам нарушений</li>
                    <li>Статус устранения</li>
                </ul>
                <a href="{{ url_for('report_violations') }}" class="btn
btn-danger">
                    <i class="bi bi-file-text"></i> Сформировать отчет
                </a>
            </div>
        </div>
    </div>
    <!-- КОНЕЦ ВСТАВКИ -->
    </div>

    <!-- Блок со стандартными отчетами из ТЗ УДАЛЕН -->
</div>
{% endblock %}
```

templates/reports/infrastructure.html

```
{% extends "base.html" %}
```

```
{% block title %}Оснащенность школ - Реестр школ Алтайского края{% endblock %}


{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-4">
        <h1><i class="bi bi-building"></i> Отчет по оснащенности школ</h1>
        <div>
            <button onclick="window.print()" class="btn btn-outline-secondary">
                <i class="bi bi-printer"></i> Печать
            </button>
            <a href="{{ url_for('export_report', report_type='infrastructure', district_id=request.args.get('district_id'), school_type_id=request.args.get('school_type_id')) }}"
               class="btn btn-success">
                <i class="bi bi-download"></i> Экспорт
            </a>
        </div>
    </div>

    <!-- Фильтры -->
    <div class="card mb-4">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-funnel"></i> Параметры отчета</h5>
        </div>
        <div class="card-body">
            <form method="GET" action="{{ url_for('report_infrastructure') }}" class="row g-3">
                <div class="col-md-4">
                    <label class="form-label">Район</label>
                    <select name="district_id" class="form-select">
```

```
                        <option value="">Все районы</option>
                        {% for district in districts %}
                        <option value="{{ district.PK_District }}"
                            {% if
request.args.get('district_id')|string == district.PK_District|string
%}selected{% endif %}>
                            {{ district.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-4">
                    <label class="form-label">Тип школы</label>
                    <select name="school_type_id" class="form-select">
                        <option value="">Все типы</option>
                        {% for type in school_types %}
                        <option value="{{ type.PK_Type_of_School }}"
                            {% if
request.args.get('school_type_id')|string ==
type.PK_Type_of_School|string %}selected{% endif %}>
                            {{ type.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-4">
                    <label class="form-label">Населенный пункт</label>
                    <select name="settlement_id" class="form-select">
                        <option value="">Все населенные пункты</option>
                        {% for settlement in settlements %}
                        <option value="{{ settlement.PK_Settlement }}"
                            {% if
request.args.get('settlement_id')|string ==
settlement.PK_Settlement|string %}selected{% endif %}>
                            {{ settlement.Type }} {{ settlement.Name }}
```

```
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-12">
                    <button type="submit" class="btn btn-primary">
                        <i class="bi bi-filter"></i> Сформировать отчет
                    </button>
                    <a href="{{ url_for('report_infrastructure') }}"
class="btn btn-outline-secondary">
                        <i class="bi bi-x-circle"></i> Сбросить
                    </a>
                </div>
            </form>
        </div>
    </div>

    <!-- Общая статистика -->
    <div class="row mb-4">
        <div class="col-md-3">
            <div class="card bg-primary text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего школ</h6>
                    <h2 class="mb-0">{{ total_schools }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-success text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">С библиотекой</h6>
                    <h2 class="mb-0">{{ with_library }}</h2>
                </div>
```

```
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-info text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">С спортзалом</h6>
                    <h2 class="mb-0">{{ with_gym }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-warning text-dark">
                <div class="card-body text-center">
                    <h6 class="card-title">С лабораториями</h6>
                    <h2 class="mb-0">{{ with_labs }}</h2>
                </div>
            </div>
        </div>
    </div>

    <!-- Статистика по инфраструктуре -->
    <div class="card mb-4">
        <div class="card-header bg-success text-white">
            <h5 class="mb-0"><i class="bi bi-bar-chart"></i>
Распределение по инфраструктуре</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>Тип инфраструктуры</th>
                            <th>Количество школ</th>
```

```
                            <th>Процент от общего</th>
                            <th>Процент школ в районе</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for item in infrastructure_stats %}
                        <tr>
                            <td><strong>{{ item.name }}</strong></td>
                            <td>{{ item.count }}</td>
                            <td>
                                <div class="progress" style="height:
20px;">
                                    <div class="progress-bar"
role="progressbar"
                                        data-width="{{ item.percentage }}"
                                        aria-valuenow="{{ item.percentage
}}"
                                        aria-valuemin="0"
                                        aria-valuemax="100">
                                        {{ "%.1f"|format(item.percentage)
}}%
                                    </div>
                                </div>
                            </td>
                            <td>
                                <small>
                                    {% for district in item.districts
%}
                                    {{ district.name }}: {{
district.percentage }}%<br>
                                    {% endfor %}
                                </small>
                            </td>
                        </tr>
                        {% endfor %}
```

```
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    <!-- Школы с детальной инфраструктурой -->
    <div class="card">
        <div class="card-header bg-info text-white">
            <h5 class="mb-0"><i class="bi bi-table"></i> Детальная
информация по школам</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Название школы</th>
                            <th>Тип школы</th>
                            <th>Населенный пункт</th>
                            <th>Инфраструктура</th>
                            <th>Объектов</th>
                            <th>Год основания</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for school in schools %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td>
                                <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}">
```

```
                                    {{ school.Official_Name }}
                                </a>
                            </td>
                            <td>{{ school.type_of_school.Name if
school.type_of_school else '' }}</td>
                            <td>
                                {% if school.settlement %}
                                {{ school.settlement.Type }} {{
school.settlement.Name }}
                                {% endif %}
                            </td>
                            <td>
                                <div class="d-flex flex-wrap gap-1">
                                    {% for infra in
school.infrastructure %}
                                    <span class="badge bg-secondary">{{
infra.Name }}</span>
                                    {% endfor %}
                                </div>
                            </td>
                            <td>
                                <span class="badge bg-info">{{
school.infrastructure|length }}</span>
                            </td>
                            <td>{{ school.Founding_Date.year if
school.Founding_Date else 'Нет данных' }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    <!-- Рекомендации -->
```

```
    {% if total_schools > 0 %}

    <div class="card mt-4">

        <div class="card-header bg-warning text-dark">

            <h5 class="mb-0"><i class="bi bi-lightbulb"></i>
Рекомендации по оснащению</h5>

        </div>

        <div class="card-body">

            <ul>

                {% set library_percentage =
(with_library/total_schools*100) if total_schools > 0 else 0 %}

                {% set gym_percentage = (with_gym/total_schools*100) if
total_schools > 0 else 0 %}

                {% set labs_percentage = (with_labs/total_schools*100)
if total_schools > 0 else 0 %}



                {% if library_percentage < 80 %}

                <li>Рекомендуется увеличить количество школ с
библиотеками (сейчас {{ "%.1f"|format(library_percentage) }}%)</li>

                {% endif %}

                {% if gym_percentage < 70 %}

                <li>Необходимо улучшить спортивную инфраструктуру
(спортзалы есть в {{ "%.1f"|format(gym_percentage) }}% школ)</li>

                {% endif %}

                {% if labs_percentage < 60 %}

                <li>Требуется оснащение лабораториями (наличие в {{
"%.1f"|format(labs_percentage) }}% школ)</li>

                {% endif %}

            </ul>

        </div>

    </div>

    {% endif %}

</div>



<script>

// Устанавливаем ширину прогресс-баров после загрузки страницы
```

```
document.addEventListener('DOMContentLoaded', function() {
    // Все прогресс-бары уже имеют правильную ширину из-за inline стилей
    // Дополнительная логика не требуется
});
</script>

{% endblock %}
```

templates/reports/programs.html

```
{% extends "base.html" %}

{% block title %}Образовательные программы - Реестр школ Алтайского края{% endblock %}

{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-4">
        <h1><i class="bi bi-book"></i> Отчет по образовательным программам</h1>
        <div>
            <button onclick="window.print()" class="btn btn-outline-secondary">
                <i class="bi bi-printer"></i> Печать
            </button>
            <a href="{{ url_for('export_schools') }}?format=excel&report=programs" class="btn btn-success">
                <i class="bi bi-download"></i> Экспорт
            </a>
        </div>
    </div>

    <!-- Параметры отчета -->
    <div class="card mb-4">
```

```
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-funnel"></i> Параметры отчета</h5>
        </div>
        <div class="card-body">
            <form method="GET" action="{{ url_for('report_programs') }}" class="row g-3">
                <div class="col-md-4">
                    <label class="form-label">Район</label>
                    <select name="district_id" class="form-select">
                        <option value="">Все районы</option>
                        {% for district in districts %}
                        <option value="{{ district.PK_District }}"
                            {% if request.args.get('district_id')|string == district.PK_District|string %}selected{% endif %}>
                            {{ district.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-4">
                    <label class="form-label">Тип школы</label>
                    <select name="school_type_id" class="form-select">
                        <option value="">Все типы</option>
                        {% for type in school_types %}
                        <option value="{{ type.PK_Type_of_School }}"
                            {% if request.args.get('school_type_id')|string == type.PK_Type_of_School|string %}selected{% endif %}>
                            {{ type.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
```

```
                <div class="col-md-4">
                    <label class="form-label">Тип программы</label>
                    <select name="program_type" class="form-select">
                        <option value="">Все типы</option>
                        <option value="основная" {% if
request.args.get('program_type') == 'основная' %}selected{% endif
%}>Основная</option>
                        <option value="дополнительная" {% if
request.args.get('program_type') == 'дополнительная' %}selected{% endif
%}>Дополнительная</option>
                    </select>
                </div>
                <div class="col-12">
                    <button type="submit" class="btn btn-primary">
                        <i class="bi bi-filter"></i> Сформировать отчет
                    </button>
                    <a href="{{ url_for('report_programs') }}"
class="btn btn-outline-secondary">
                        <i class="bi bi-x-circle"></i> Сбросить
                    </a>
                </div>
            </form>
        </div>
    </div>

    <!-- Общая статистика -->
    <div class="row mb-4">
        <div class="col-md-4">
            <div class="card bg-primary text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего программ</h6>
                    <h2 class="mb-0">{{ programs|length }}</h2>
                </div>
            </div>
```

```
        </div>
        <div class="col-md-4">
            <div class="card bg-success text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Основные программы</h6>
                    <h2 class="mb-0">{{
program_stats|selectattr('program.Type', 'equalto',
'основная')|list|length }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-4">
            <div class="card bg-info text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Дополнительные
программы</h6>
                    <h2 class="mb-0">{{
program_stats|selectattr('program.Type', 'equalto',
'дополнительная')|list|length }}</h2>
                </div>
            </div>
        </div>
    </div>

    <!-- Распределение по типам программ -->
    <div class="card mb-4">
        <div class="card-header bg-success text-white">
            <h5 class="mb-0"><i class="bi bi-pie-chart"></i>
Распределение по типам программ</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
```

```
                        <th>Тип программы</th>
                        <th>Количество</th>
                        <th>Процент</th>
                    </tr>
                </thead>
                <tbody>
                    {% for stat in type_stats %}
                    <tr>
                        <td><strong>{{ stat.type }}</strong></td>
                        <td>{{ stat.count }}</td>
                        <td>
                            {% set percentage = (stat.count /
programs|length * 100) if programs|length > 0 else 0 %}
                            <div class="progress" style="height:
20px;">
                                <div class="progress-bar"
role="progressbar"
                                     data-width="{{
percentage|round(1) }}"
                                     aria-valuenow="{{
percentage|round(1) }}"
                                     aria-valuemin="0"
aria-valuemax="100">
                                    {{ "%.1f"|format(percentage)
}}%
                                </div>
                            </div>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
```

```
    <!-- Статистика по программам -->
    <div class="card">
        <div class="card-header bg-info text-white">
            <h5 class="mb-0"><i class="bi bi-table"></i> Статистика по программам</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Код программы</th>
                            <th>Название программы</th>
                            <th>Тип программы</th>
                            <th>Количество школ</th>
                            <th>Процент школ</th>
                            <th>Пример школ</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for stat in program_stats %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td><code>{{ stat.program.Code_Designation }}</code></td>
                            <td><strong>{{ stat.program.Name }}</strong></td>
                            <td>
                                <span class="badge {% if stat.program.Type == 'основная' %}bg-primary{% else %}bg-success{% endif %}">
                                    {{ stat.program.Type }}
```

```
                                </span>
                            </td>
                            <td>{{ stat.schools_count }}</td>
                            <td>
                                <div class="progress" style="height:
20px;">
                                    <div class="progress-bar"
role="progressbar"
                                         data-width="{{ stat.percentage
}}"
                                         aria-valuenow="{{
stat.percentage }}"
                                         aria-valuemin="0"
aria-valuemax="100">
                                        {{
"%.1f"|format(stat.percentage) }}%
                                    </div>
                                </div>
                            </td>
                            <td>
                                <small>
                                    {% for school in stat.schools[:3]
%}
                                    {{ school.Official_Name[:20]
}}...<br>
                                    {% endfor %}
                                    {% if stat.schools_count > 3 %}
                                    <span class="text-muted">и еще {{
stat.schools_count - 3 }} школ</span>
                                    {% endif %}
                                </small>
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
```

```
            </div>
        </div>
    </div>
</div>


<script>
document.addEventListener('DOMContentLoaded', function() {
    // Устанавливаем ширину прогресс-баров из data-width

document.querySelectorAll('.progress-bar[data-width]').forEach(function
(el) {
        const width = parseFloat(el.getAttribute('data-width'));
        if (!isNaN(width)) {
            el.style.width = width + '%';
        }
    });
});
</script>
{% endblock %}
```

templates/reports/staff.html

```
{% extends "base.html" %}


{% block title %}Кадровый состав - Реестр школ Алтайского края{%
endblock %}


{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center
mb-4">
        <h1><i class="bi bi-people"></i> Отчет по кадровому
составу</h1>
        <div>
            <button onclick="window.print()" class="btn
btn-outline-secondary">
```

```
                <i class="bi bi-printer"></i> Печать
            </button>
            <button class="btn btn-success"
onclick="exportReport('staff')">
                <i class="bi bi-download"></i> Экспорт
            </button>
        </div>
    </div>

    <!-- Параметры отчета -->
    <div class="card mb-4">
        <div class="card-header bg-primary text-white">
            <h5 class="mb-0"><i class="bi bi-gear"></i> Параметры
отчета</h5>
        </div>
        <div class="card-body">
            <form method="GET" action="{{ url_for('report_staff') }}"
class="row g-3">
                <div class="col-md-3">
                    <label class="form-label">Район</label>
                    <select name="district_id" class="form-select">
                        <option value="">Все районы</option>
                        {% for district in districts %}
                        <option value="{{ district.PK_District }}"
                            {% if
request.args.get('district_id')|string == district.PK_District|string
%}selected{% endif %}>
                            {{ district.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label">Тип школы</label>
```

```
                    <select name="school_type_id" class="form-select">
                        <option value="">Все типы</option>
                        {% for type in school_types %}
                        <option value="{{ type.PK_Type_of_School }}"
                            {% if
request.args.get('school_type_id')|string ==
type.PK_Type_of_School|string %}selected{% endif %}>
                            {{ type.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label">Должность</label>
                    <select name="position" class="form-select">
                        <option value="">Все должности</option>
                        <option value="директор" {% if
request.args.get('position') == 'директор' %}selected{% endif
%}>Директор</option>
                        <option value="завуч" {% if
request.args.get('position') == 'завуч' %}selected{% endif
%}>Завуч</option>
                        <option value="учитель" {% if
request.args.get('position') == 'учитель' %}selected{% endif
%}>Учитель</option>
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label">Стаж от (лет)</label>
                    <input type="number" name="min_experience"
class="form-control"
                           value="{{ request.args.get('min_experience',
'0') }}" min="0" max="50">
                </div>
                <div class="col-12">
                    <button type="submit" class="btn btn-primary">
```

```
                        <i class="bi bi-filter"></i> Сформировать отчет
                    </button>
                    <a href="{{ url_for('report_staff') }}" class="btn
btn-outline-secondary">
                        <i class="bi bi-x-circle"></i> Сбросить
                    </a>
                </div>
            </form>
        </div>
    </div>


    <!-- Общая статистика -->
    <div class="row mb-4">
        <div class="col-md-3">
            <div class="card bg-primary text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего сотрудников</h6>
                    <h2 class="mb-0">{{ total_employees }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-success text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Директоров</h6>
                    <h2 class="mb-0">{{ directors_count }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-info text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Завучей</h6>
```

```
                    <h2 class="mb-0">{{ deputies_count }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-warning text-dark">
                <div class="card-body text-center">
                    <h6 class="card-title">Учителей</h6>
                    <h2 class="mb-0">{{ teachers_count }}</h2>
                </div>
            </div>
        </div>
    </div>


    <!-- Распределение по должностям -->
    <div class="card mb-4">
        <div class="card-header bg-success text-white">
            <h5 class="mb-0"><i class="bi bi-pie-chart"></i>
Распределение по должностям</h5>
        </div>
        <div class="card-body">
            <div class="row">
                <div class="col-md-6">
                    <div class="table-responsive">
                        <table class="table table-hover">
                            <thead>
                                <tr>
                                    <th>Должность</th>
                                    <th>Количество</th>
                                    <th>Процент</th>
                                </tr>
                            </thead>
                            <tbody>
```

```
                                {% for stat in position_stats %}
                                <tr>
                                    <td><strong>{{ stat.position
}}</strong></td>
                                    <td>{{ stat.count }}</td>
                                    <td>
                                        <div class="progress"
style="height: 20px;">
                                            <div class="progress-bar"
role="progressbar"
                                                data-width="{{
stat.percentage }}"
                                                aria-valuenow="{{
stat.percentage }}"
                                                aria-valuemin="0"
                                                aria-valuemax="100">
                                                {{
"%.1f"|format(stat.percentage) }}%
                                            </div>
                                    </div>
                                    </td>
                                </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                </div>
                <div class="col-md-6">
                    <!-- График можно добавить позже -->
                    <div class="text-center py-5">
                        <i class="bi bi-bar-chart display-1
text-muted"></i>
                        <p class="mt-3">График распределения по
должностям</p>
                        <small class="text-muted">(Визуализация будет
добавлена в следующей версии)</small>
```

```
                    </div>
                </div>
            </div>
        </div>

    <!-- Статистика по стажу -->
    <div class="card mb-4">
        <div class="card-header bg-info text-white">
            <h5 class="mb-0"><i class="bi bi-graph-up"></i> Статистика
по стажу работы</h5>
        </div>
        <div class="card-body">
            <div class="row">
                <div class="col-md-4">
                    <div class="card bg-light">
                        <div class="card-body text-center">
                            <h6>Средний стаж</h6>
                            <h3>{{ avg_experience }} лет</h3>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card bg-light">
                        <div class="card-body text-center">
                            <h6>Максимальный стаж</h6>
                            <h3>{{ max_experience }} лет</h3>
                        </div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="card bg-light">
                        <div class="card-body text-center">
```

```
                        <h6>Минимальный стаж</h6>
                        <h3>{{ min_experience }} лет</h3>
                    </div>
                </div>
            </div>

            <div class="mt-4">
                <h6>Распределение по стажу:</h6>
                <div class="table-responsive">
                    <table class="table table-sm">
                        <thead>
                            <tr>
                                <th>Стаж (лет)</th>
                                <th>Количество сотрудников</th>
                                <th>Процент</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for range in experience_ranges %}
                            <tr>
                                <td>{{ range.range }}</td>
                                <td>{{ range.count }}</td>
                                <td>
                                <div class="progress" style="height:
15px;">
                                <div class="progress-bar"
role="progressbar"
                                    data-width="{{ range.percentage }}"
                                    aria-valuenow="{{ range.percentage
}}"
                                    aria-valuemin="0"
                                    aria-valuemax="100">
```

```
                                </div>
                            </div>
                                </td>
                            </tr>
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>


    <!-- Детальная информация -->
    <div class="card">
        <div class="card-header bg-dark text-white">
            <h5 class="mb-0"><i class="bi bi-table"></i> Детальная
информация по сотрудникам</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>ФИО</th>
                            <th>Должность</th>
                            <th>Стаж (лет)</th>
                            <th>Школа</th>
                            <th>Населенный пункт</th>
                            <th>Квалификация</th>
                        </tr>
                    </thead>
                    <tbody>
```

```
                        {% for employee in employees %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td><strong>{{ employee.Full_Name
}}</strong></td>
                            <td>
                                {% set position_lower =
employee.Position.lower() %}
                                <span class="badge
                                    {% if 'директор' in position_lower
%}bg-danger
                                    {% elif 'завуч' in position_lower
or 'заместитель' in position_lower %}bg-warning text-dark
                                    {% else %}bg-info{% endif %}">
                                    {{ employee.Position }}
                                </span>
                            </td>
                            <td>{{ employee.Experience_Years or 'Нет
данных' }}</td>
                            <td>
                                {% if employee.schools %}
                                    {{
employee.schools[0].Official_Name }}
                                {% else %}
                                    Не назначен
                                {% endif %}
                            </td>
                            <td>
                                {% if employee.schools and
employee.schools[0].settlement %}
                                    {{
employee.schools[0].settlement.Type }} {{
employee.schools[0].settlement.Name }}
                                {% endif %}
                            </td>
                            <td>
```

```
                                <small>{{ (employee.Qualifications[:50] if employee.Qualifications else '') }}{% if employee.Qualifications and employee.Qualifications|length > 50 %}...{% endif %}</small>
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
{% endblock %}

{% block scripts %}
<script>
function exportReport(reportType) {
    const params = new URLSearchParams(window.location.search);
    params.set('format', 'excel');
    window.location.href = `/export/report?${params.toString()}&report_type=${reportType}`;
}

// Инициализация прогресс-баров
document.addEventListener('DOMContentLoaded', function() {
    // Устанавливаем ширину прогресс-баров
    document.querySelectorAll('.progress-bar[style*="width:"]').forEach(function(el) {
        const style = el.getAttribute('style');
        if (style.includes('width:')) {
            // Уже установлено
        }
    });
```

```
});

</script>

{% endblock %}
```

templates/reports/stduent.html

```
{% extends "base.html" %}



{% block title %}Статистика по учащимся - Реестр школ Алтайского края{%
endblock %}



{% block content %}

<div class="container">

    <div class="d-flex justify-content-between align-items-center
mb-4">

        <h1><i class="bi bi-people"></i> Статистика по количеству
учащихся</h1>

        <div>

            <button onclick="window.print()" class="btn
btn-outline-secondary">

                <i class="bi bi-printer"></i> Печать

            </button>

            <a href="{{ url_for('export_schools') }}?format=excel"
class="btn btn-success">

                <i class="bi bi-download"></i> Экспорт

            </a>

        </div>

    </div>



    <!-- Фильтры -->

    <div class="card mb-4">

        <div class="card-header bg-primary text-white">

            <h5 class="mb-0"><i class="bi bi-funnel"></i> Фильтры
отчета</h5>

        </div>

        <div class="card-body">
```

```
            <form method="GET" action="{{ url_for('report_students')
}}" class="row g-3">

                <div class="col-md-4">

                    <label class="form-label">Район</label>

                    <select name="district_id" class="form-select">

                        <option value="">Все районы</option>

                        {% for district in districts %}

                        <option value="{{ district.PK_District }}"

                            {% if
request.args.get('district_id')|string == district.PK_District|string
%}selected{% endif %}>

                            {{ district.Name }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="col-md-4">

                    <label class="form-label">Тип школы</label>

                    <select name="type_id" class="form-select">

                        <option value="">Все типы</option>

                        {% for type in school_types %}

                        <option value="{{ type.PK_Type_of_School }}"

                            {% if request.args.get('type_id')|string ==
type.PK_Type_of_School|string %}selected{% endif %}>

                            {{ type.Name }}

                        </option>

                        {% endfor %}

                    </select>

                </div>

                <div class="col-md-2">

                    <label class="form-label">Год от</label>

                    <input type="number" name="min_year"
class="form-control"

                           value="{{ request.args.get('min_year', 2000)
}}" min="1900" max="2026">
```

```
                </div>
                <div class="col-md-2">
                    <label class="form-label">Год до</label>
                    <input type="number" name="max_year"
class="form-control"
                           value="{{ request.args.get('max_year', 2026)
}}" min="1900" max="2026">
                </div>
                <div class="col-12">
                    <button type="submit" class="btn btn-primary">
                        <i class="bi bi-filter"></i> Применить фильтры
                    </button>
                    <a href="{{ url_for('report_students') }}"
class="btn btn-outline-secondary">
                        <i class="bi bi-x-circle"></i> Сбросить
                    </a>
                </div>
            </form>
        </div>
    </div>

    <!-- Общая статистика -->
    <div class="row mb-4">
        <div class="col-md-3">
            <div class="card bg-primary text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Всего школ</h6>
                    <h2 class="mb-0">{{ total_schools }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-success text-white">
```

```
                <div class="card-body text-center">
                    <h6 class="card-title">Всего учащихся</h6>
                    <h2 class="mb-0">{{ total_students }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-info text-white">
                <div class="card-body text-center">
                    <h6 class="card-title">Среднее на школу</h6>
                    <h2 class="mb-0">{{
avg_students_per_school|round(1) }}</h2>
                </div>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card bg-warning text-dark">
                <div class="card-body text-center">
                    <h6 class="card-title">Самая большая</h6>
                    <h4 class="mb-0">{{ max_students }}</h4>
                    <small>{% if max_school %}{{
max_school.Official_Name[:20] }}...{% else %}Нет данных{% endif
%}</small>
                </div>
            </div>
        </div>
    </div>


    <!-- Статистика по районам -->
    {% if districts_stats %}
    <div class="card mb-4">
        <div class="card-header bg-success text-white">
            <h5 class="mb-0"><i class="bi bi-map"></i> Статистика по
районам</h5>
```

```
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>Район</th>
                            <th>Количество школ</th>
                            <th>Всего учащихся</th>
                            <th>Среднее на школу</th>
                            <th>Процент от общего</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for stat in districts_stats %}
                        <tr>
                            <td><strong>{{ stat.district
}}</strong></td>
                            <td>{{ stat.schools_count }}</td>
                            <td>{{ stat.total_students }}</td>
                            <td>{{ stat.avg_students }}</td>
                            <td>
                                {% if stat.percentage > 0 %}
                                <div class="progress" style="height:
20px;">
                                    <div class="progress-bar"
role="progressbar"
                                         data-width="{{ stat.percentage
}}"
                                         aria-valuenow="{{
stat.percentage }}"
                                         aria-valuemin="0"
aria-valuemax="100">
                                        {{
"%.1f"|format(stat.percentage) }}%
```

```
                                    </div>
                                </div>
                                {% else %}
                                <span class="text-muted">0%</span>
                                {% endif %}
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    {% endif %}


    <!-- Статистика по типам школ -->
    {% if types_stats %}
    <div class="card mb-4">
        <div class="card-header bg-info text-white">
            <h5 class="mb-0"><i class="bi bi-building"></i> Статистика
по типам школ</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>Тип школы</th>
                            <th>Количество школ</th>
                            <th>Всего учащихся</th>
                            <th>Среднее на школу</th>
                        </tr>
                    </thead>
```

```
                    <tbody>
                        {% for stat in types_stats %}
                        <tr>
                            <td><strong>{{ stat.type }}</strong></td>
                            <td>{{ stat.schools_count }}</td>
                            <td>{{ stat.total_students }}</td>
                            <td>{{ stat.avg_students }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    {% endif %}


    <!-- Динамика набора -->
    {% if yearly_stats %}
    <div class="card mb-4">
        <div class="card-header bg-warning text-dark">
            <h5 class="mb-0"><i class="bi bi-graph-up"></i> Динамика набора учащихся</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>Год основания</th>
                            <th>Количество школ</th>
                            <th>Среднее количество учащихся</th>
                        </tr>
                    </thead>
```

```
                    <tbody>
                        {% for year, data in yearly_stats.items()|sort
%}
                        <tr>
                            <td>{{ year }}</td>
                            <td>{{ data.count }}</td>
                            <td>{{ "%.1f"|format(data.avg_students)
}}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    {% endif %}

    <!-- Топ школ по количеству учащихся -->
    {% if top_schools %}
    <div class="card">
        <div class="card-header bg-dark text-white">
            <h5 class="mb-0"><i class="bi bi-trophy"></i> Топ-10 школ
по количеству учащихся</h5>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Название школы</th>
                            <th>Тип школы</th>
                            <th>Населенный пункт</th>
```

```
                            <th>Количество учащихся</th>
                            <th>Год основания</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for school in top_schools %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td>
                                <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}">
                                    {{ school.Official_Name }}
                                </a>
                            </td>
                            <td>{{ school.type_of_school.Name if
school.type_of_school else '' }}</td>
                            <td>
                                {% if school.settlement %}
                                    {{ school.settlement.Type }} {{
school.settlement.Name }}
                                {% endif %}
                            </td>
                            <td><strong>{{ school.Number_of_Students or
0 }}</strong></td>
                            <td>{{ school.Founding_Date.year if
school.Founding_Date else 'Нет данных' }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    {% endif %}
```

```
</div>


<script>
document.addEventListener('DOMContentLoaded', function() {
    // Устанавливаем ширину прогресс-баров из data-width

document.querySelectorAll('.progress-bar[data-width]').forEach(function
(el) {
        const width = parseFloat(el.getAttribute('data-width'));
        if (!isNaN(width)) {
            el.style.width = width + '%';
        }
    });
});
</script>
{% endblock %}
```

templates/reports/violations.html

```
<!-- templates/reports/violations.html -->
{% extends "base.html" %}


{% block title %}Отчет по нарушениям Рособрнадзора - Реестр школ
Алтайского края{% endblock %}


{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center
mb-4">
        <h1><i class="bi bi-exclamation-triangle"></i> Отчет по
нарушениям Рособрнадзора</h1>
        <div>
            <button onclick="window.print()" class="btn
btn-outline-secondary">
                <i class="bi bi-printer"></i> Печать
            </button>
```

```
            <a href="{{ url_for('export_report',
report_type='violations') }}" class="btn btn-success">
                <i class="bi bi-download"></i> Экспорт
            </a>
        </div>
    </div>

    <!-- Фильтры -->
    <div class="card mb-4">
        <div class="card-header bg-danger text-white">
            <h5 class="mb-0"><i class="bi bi-funnel"></i> Параметры
отчета</h5>
        </div>
        <div class="card-body">
            <form method="GET" action="{{ url_for('report_violations')
}}" class="row g-3">
                <div class="col-md-3">
                    <label class="form-label">Район</label>
                    <select name="district_id" class="form-select">
                        <option value="">Все районы</option>
                        {% for district in districts %}
                        <option value="{{ district.PK_District }}"
                                {% if
request.args.get('district_id')|string == district.PK_District|string
%}selected{% endif %}>
                            {{ district.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label">Тип школы</label>
                    <select name="school_type_id" class="form-select">
                        <option value="">Все типы</option>
```

```
                        {% for type in school_types %}
                        <option value="{{ type.PK_Type_of_School }}"
                                {% if
request.args.get('school_type_id')|string ==
type.PK_Type_of_School|string %}selected{% endif %}>
                            {{ type.Name }}
                        </option>
                        {% endfor %}
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label">Статус нарушений</label>
                    <select name="status" class="form-select">
                        <option value="all" {% if
request.args.get('status') == 'all' %}selected{% endif %}>Все
проверки</option>
                        <option value="with_violations" {% if
request.args.get('status') == 'with_violations' %}selected{% endif %}>С
нарушениями</option>
                        <option value="active" {% if
request.args.get('status') == 'active' %}selected{% endif %}>Не
устранены</option>
                        <option value="resolved" {% if
request.args.get('status') == 'resolved' %}selected{% endif
%}>Устранены</option>
                    </select>
                </div>
                <div class="col-md-3">
                    <label class="form-label"> </label>
                    <button type="submit" class="btn btn-danger w-100">
                        <i class="bi bi-filter"></i> Сформировать отчет
                    </button>
                </div>
            </form>
        </div>
    </div>
```

```
<!-- Статистика -->
<div class="row mb-4">
    <div class="col-md-3">
        <div class="card bg-danger text-white">
            <div class="card-body text-center">
                <h6 class="card-title">Всего проверок</h6>
                <h2 class="mb-0">{{ total_inspections }}</h2>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card bg-warning text-dark">
            <div class="card-body text-center">
                <h6 class="card-title">С нарушениями</h6>
                <h2 class="mb-0">{{ with_violations }}</h2>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card bg-success text-white">
            <div class="card-body text-center">
                <h6 class="card-title">Устранены</h6>
                <h2 class="mb-0">{{ resolved }}</h2>
            </div>
        </div>
    </div>
    <div class="col-md-3">
        <div class="card bg-info text-white">
            <div class="card-body text-center">
                <h6 class="card-title">Не устранены</h6>
                <h2 class="mb-0">{{ active }}</h2>
```

```
                </div>
            </div>
        </div>
    </div>

    <!-- Таблица нарушений -->
    <div class="card">
        <div class="card-header bg-dark text-white">
            <h5 class="mb-0"><i class="bi bi-table"></i> Список
нарушений</h5>
        </div>
        <div class="card-body">
            {% if inspections %}
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Дата проверки</th>
                            <th>Номер предписания</th>
                            <th>Школа</th>
                            <th>Тип нарушения</th>
                            <th>Результат</th>
                            <th>Статус</th>
                            <th>Дата устранения</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for inspection in inspections %}
                        <tr>
                            <td>{{ loop.index }}</td>
                            <td>{{ inspection.Date.strftime('%d.%m.%Y')
}}</td>
```

```
                            <td><strong>{{
inspection.Prescription_Number or 'Без номера' }}</strong></td>

                            <td>

                                <a href="{{ url_for('school_detail',
school_id=inspection.PK_School) }}">

                                    {{
inspection.school.Official_Name[:50] }}{% if
inspection.school.Official_Name|length > 50 %}...{% endif %}

                                </a>

                            </td>

                            <td>{{ inspection.violation_type[:50] if
inspection.violation_type else 'Не указан' }}</td>

                            <td>{{ inspection.Result[:100] if
inspection.Result else 'Нет описания' }}{% if inspection.Result and
inspection.Result|length > 100 %}...{% endif %}</td>

                            <td>

                                {% if inspection.has_violations %}

                                    {% if inspection.is_resolved %}

                                        <span class="badge
bg-success">Устранено</span>

                                    {% else %}

                                        <span class="badge bg-warning
text-dark">Не устранено</span>

                                    {% endif %}

                                {% else %}

                                    <span class="badge
bg-info">Нарушений нет</span>

                                {% endif %}

                            </td>

                            <td>

                                {% if inspection.resolution_date %}

                                    {{
inspection.resolution_date.strftime('%d.%m.%Y') }}

                                {% else %}

                                    <span class="text-muted">-</span>

                                {% endif %}
```

```
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
            {% else %}
            <div class="alert alert-info">
                <i class="bi bi-info-circle"></i>
                По выбранным критериям нарушения не найдены.
            </div>
            {% endif %}
        </div>
    </div>
</div>
{% endblock %}
```

templates/schools/detail.html

```
{% extends "base.html" %}

{% block title %}{{ school.Official_Name }} - Реестр школ Алтайского
края{% endblock %}

{% block head %}
<style>
    .school-header {
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        color: white;
        padding: 2rem 0;
        margin-bottom: 2rem;
        border-radius: 10px;
    }
```

```
    .info-card {
        transition: all 0.3s ease;
    }

    .info-card:hover {
        transform: translateY(-3px);
        box-shadow: 0 5px 15px rgba(0,0,0,0.1);
    }

    .rating-large {
        font-size: 1.5rem;
    }
</style>
{% endblock %}

{% block content %}
<!-- Заголовок школы -->
<div class="school-header">
    <div class="container">
        <div class="row align-items-center">
            <div class="col-md-8">
                <h1 class="display-5 fw-bold mb-2">{{ school.Official_Name }}</h1>

                <div class="d-flex flex-wrap gap-2 mb-3">
                    <span class="badge bg-light text-dark">
                        <i class="bi bi-building"></i> {{ school.type_of_school.Name }}
                    </span>
                    <span class="badge bg-light text-dark">
                        <i class="bi bi-geo-alt"></i> {{ school.settlement.Type }} {{ school.settlement.Name }}
                    </span>
```

```
                    {% if school.Number_of_Students %}
                    <span class="badge bg-light text-dark">
                        <i class="bi bi-people"></i> {{
school.Number_of_Students }} учащихся
                    </span>
                    {% endif %}
                </div>

                <!-- Рейтинг -->
                <div class="d-flex align-items-center">
                    <div class="rating-large me-3">
                        {{ render_star_rating(avg_rating)|safe }}
                    </div>
                    <div>
                        <div class="h4 mb-0">{{ avg_rating|round(1)
}}/5.0</div>
                        <small>{{ review_count }} {{
pluralize(review_count, 'отзыв', 'отзыва', 'отзывов') }}</small>
                    </div>
                </div>
            </div>

            <div class="col-md-4 text-end">
                {% if current_user.is_authenticated and
can_edit_school(school.PK_School) %}
                <a href="{{ url_for('edit_school',
school_id=school.PK_School) }}"
                   class="btn btn-light btn-lg">
                    <i class="bi bi-pencil"></i> Редактировать
                </a>
                 <a href="{{ url_for('school_history',
school_id=school.PK_School) }}"
                class="btn btn-info mt-2">
                    <i class="bi bi-clock-history"></i> История
изменений
```

```
                </a>
                {% endif %}
            </div>
        </div>
    </div>
</div>

<div class="container">
    <div class="row">
        <!-- Левая колонка - основная информация -->
        <div class="col-lg-8">
            <!-- Контактная информация -->
            <div class="card mb-4 info-card">
                <div class="card-header bg-primary text-white">
                    <h5 class="mb-0"><i class="bi bi-telephone"></i>
Контактная информация</h5>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col-md-6 mb-3">
                            <h6><i class="bi bi-geo-alt
text-primary"></i> Адрес</h6>
                            <p class="mb-0">{{ school.Legal_Adress
}}</p>
                        </div>

                        <div class="col-md-6 mb-3">
                            <h6><i class="bi bi-telephone
text-primary"></i> Телефон</h6>
                            <p class="mb-0">{{ school.Phone }}</p>
                        </div>

                        {% if school.Email %}
                        <div class="col-md-6 mb-3">
```

```
                        <h6><i class="bi bi-envelope
text-primary"></i> Email</h6>
                        <p class="mb-0">{{ school.Email }}</p>
                    </div>
                    {% endif %}

                    {% if school.Website %}
                    <div class="col-md-6 mb-3">
                        <h6><i class="bi bi-globe
text-primary"></i> Веб-сайт</h6>
                        <p class="mb-0">
                            <a href="{{ school.Website }}"
target="_blank">{{ school.Website }}</a>
                        </p>
                    </div>
                    {% endif %}
                </div>
            </div>
        </div>

        <!-- Основные сведения -->
        <div class="card mb-4 info-card">
            <div class="card-header bg-success text-white">
                <h5 class="mb-0"><i class="bi bi-info-circle"></i>
Основные сведения</h5>
            </div>
            <div class="card-body">
                <div class="row">
                    {% if school.Founding_Date %}
                    <div class="col-md-6 mb-3">
                        <h6>Дата основания</h6>
                        <p class="mb-0">{{
school.Founding_Date.strftime('%d.%m.%Y') }}</p>
                    </div>
```

```
                        {% endif %}


                        {% if school.Number_of_Students %}
                        <div class="col-md-6 mb-3">
                            <h6>Количество учащихся</h6>
                            <p class="mb-0">{{
school.Number_of_Students }}</p>
                        </div>
                        {% endif %}


                        {% if school.type_of_school %}
                        <div class="col-md-6 mb-3">
                            <h6>Тип образовательной организации</h6>
                            <p class="mb-0">{{
school.type_of_school.Name }}</p>
                        </div>
                        {% endif %}


                        {% if school.settlement %}
                        <div class="col-md-6 mb-3">
                            <h6>Место нахождения</h6>
                            <p class="mb-0">
                                {{ school.settlement.district.Name }},
                                {{ school.settlement.Type }} {{
school.settlement.Name }}
                            </p>
                        </div>
                            {% endif %}
                    </div>


                    {% if school.License %}
                    <div class="mt-3">
                        <h6>Лицензия</h6>
```

```
                    <p class="mb-0">{{ school.License }}</p>
                </div>
                {% endif %}

                {% if school.Accreditation %}
                <div class="mt-3">
                    <h6>Аккредитация</h6>
                    <p class="mb-0">{{ school.Accreditation }}</p>
                </div>
                {% endif %}
            </div>
        </div>

        <!-- Инфраструктура -->
        {% if school.infrastructure %}
        <div class="card mb-4 info-card">
            <div class="card-header bg-info text-white">
                <h5 class="mb-0"><i class="bi bi-building"></i>
Инфраструктура</h5>
            </div>
            <div class="card-body">
                <div class="row">
                    {% for infra in school.infrastructure %}
                    <div class="col-md-6 mb-2">
                        <div class="form-check">
                            <input class="form-check-input"
type="checkbox" checked disabled>
                            <label class="form-check-label">
                                {{ infra.Name }}
                            </label>
                        </div>
                    </div>
                    {% endfor %}
```

```
                    </div>

                </div>

            </div>

            {% endif %}


            <!-- Отзывы -->

            <div class="card mb-4 info-card">

                <div class="card-header bg-warning text-dark">

                    <h5 class="mb-0"><i class="bi
bi-chat-left-text"></i> Отзывы ({{ review_count }})</h5>

                </div>

                <div class="card-body">

                    {% if reviews %}

                    {% for review in reviews %}

                    <div class="mb-3 pb-3 border-bottom">

                        <div class="d-flex justify-content-between
mb-2">

                            <div>

                                <strong>{{ review.Author }}</strong>

                                <small class="text-muted ms-2">{{
review.Date.strftime('%d.%m.%Y') }}</small>

                            </div>

                            <div class="text-warning">

                                {{
render_star_rating(review.Rating)|safe }}

                            </div>

                        </div>

                        <p class="mb-0">{{ review.Text }}</p>


                        {% if current_user.has_role('school_admin') %}

                        <form method="POST"

                            action="{{ url_for('delete_review',
review_id=review.PK_Review) }}"

                            class="d-inline"
```

```
                            onsubmit="return confirm('Удалить этот
отзыв?')">
                            <button type="submit" class="btn btn-sm
btn-outline-danger">
                                <i class="bi bi-trash"></i> Удалить
                            </button>
                        </form>
                    {% endif %}
                    </div>
                    {% endfor %}
                    {% else %}
                    <div class="text-center py-4">
                        <i class="bi bi-chat-left-text display-1
text-muted"></i>
                        <h5 class="mt-3">Пока нет отзывов</h5>
                        <p class="text-muted">Будьте первым, кто
оставит отзыв об этой школе</p>
                    </div>
                    {% endif %}

                    <!-- Форма добавления отзыва -->
                    {% if current_user.is_authenticated %}
                    <div class="mt-4">
                        <h6>Добавить отзыв</h6>
                        <form method="POST" action="{{
url_for('add_review', school_id=school.PK_School) }}">
                            {{ form.hidden_tag() }}

                            <div class="mb-3">
                                <label
class="form-label">Оценка</label>
                                <div class="rating-input mb-2">
                                    {% for choice in
form.rating.choices %}
```

```
                                    <div class="form-check
form-check-inline">
                                        <input class="form-check-input"
type="radio"
                                               name="{{
form.rating.name }}"
                                               id="rating{{ choice[0]
}}"
                                               value="{{ choice[0] }}"
                                               {% if loop.first
%}checked{% endif %}>
                                        <label class="form-check-label"
for="rating{{ choice[0] }}">
                                            {{ choice[1] }}
                                        </label>
                                    </div>
                                    {% endfor %}
                                </div>
                                {% if form.rating.errors %}
                                <div class="text-danger small">
                                    {% for error in form.rating.errors
%}
                                    {{ error }}
                                    {% endfor %}
                                </div>
                                {% endif %}
                            </div>

                            <div class="mb-3">
                                <label class="form-label">Текст
отзыва</label>
                                {{ form.text(class="form-control",
rows="3", placeholder="Напишите ваш отзыв...") }}
                                {% if form.text.errors %}
                                <div class="text-danger small">
                                    {% for error in form.text.errors %}
```

```
                                {{ error }}
                                {% endfor %}
                            </div>
                            {% endif %}
                        </div>

                        <button type="submit" class="btn
btn-warning">
                            <i class="bi bi-send"></i> Отправить
отзыв
                        </button>
                    </form>
                </div>
                {% else %}
                <div class="alert alert-info">
                    <i class="bi bi-info-circle"></i>
                    Чтобы оставить отзыв, пожалуйста,
                    <a href="{{ url_for('login') }}">войдите в
систему</a>
                </div>
                {% endif %}
            </div>
        </div>
        <!-- Конец левой колонки -->

        <!-- Правая колонка - дополнительная информация -->
        <div class="col-lg-4">
            <!-- Карта -->
            <div class="card mb-4 info-card">
                <div class="card-header bg-danger text-white">
                    <h5 class="mb-0"><i class="bi bi-geo-alt"></i>
Местоположение</h5>
                </div>
```

```
                <div class="card-body text-center py-4">
                    <div class="mb-3">
                        <i class="bi bi-geo-alt-fill display-4
text-danger"></i>
                    </div>
                    <p class="mb-4">
                        <strong>Адрес:</strong><br>
                        {{ school.Legal_Adress }}
                    </p>
                    <a href="https://yandex.ru/maps/?text={{
school.Legal_Adress|urlencode }}"
                       target="_blank" class="btn btn-danger w-100">
                        <i class="bi bi-map"></i> Открыть в
Яндекс.Картах
                    </a>
                </div>
            </div>


            <!-- Сотрудники -->
            {% if employees %}
            <div class="card mb-4 info-card">
                <div class="card-header bg-secondary text-white">
                    <h5 class="mb-0"><i class="bi bi-people"></i>
Сотрудники</h5>
                </div>
                <div class="card-body">
                    <div class="list-group list-group-flush">
                        {% for employee in employees %}
                        <div class="list-group-item">
                            <div class="d-flex w-100
justify-content-between">
                                <h6 class="mb-1">{{ employee.Full_Name
}}</h6>
                                <small>{{ employee.Position }}</small>
                            </div>
```

```
                            {% if employee.Qualifications %}
                            <p class="mb-1 small text-muted">{{
employee.Qualifications|truncate(50) }}</p>
                            {% endif %}
                        </div>
                        {% endfor %}
                    </div>

                    {% if employees|length > 10 %}
                    <div class="text-center mt-3">
                        <a href="#" class="btn btn-sm
btn-outline-secondary">
                            Показать всех сотрудников ({{
school.employees|length }})
                        </a>
                    </div>
                    {% endif %}
                </div>
            </div>
            {% endif %}

            <!-- Программы -->
            {% if programs %}
            <div class="card mb-4 info-card">
                <div class="card-header bg-dark text-white">
                    <h5 class="mb-0"><i class="bi bi-book"></i>
Образовательные программы</h5>
                </div>
                <div class="card-body">
                    <div class="list-group list-group-flush">
                        {% for program in programs %}
                        <div class="list-group-item">
                            <div class="d-flex w-100
justify-content-between">
```

```
                                <h6 class="mb-1">{{ program.Name
}}</h6>
                                <small class="text-muted">{{
program.Type }}</small>
                            </div>
                            <p class="mb-1 small">
                                <code>{{ program.Code_Designation
}}</code>
                            </p>
                        </div>
                        {% endfor %}
                    </div>

                    {% if programs|length > 10 %}
                    <div class="text-center mt-3">
                        <a href="#" class="btn btn-sm
btn-outline-dark">
                            Все программы ({{ school.programs|length
}})
                        </a>
                    </div>
                    {% endif %}
                </div>
            </div>
            {% endif %}

            <!-- Действия -->
            <div class="card info-card">
                <div class="card-header bg-light">
                    <h5 class="mb-0"><i class="bi bi-lightning"></i>
Быстрые действия</h5>
                </div>
                <div class="card-body">
                    <div class="d-grid gap-2">
```

```
                        <a href="tel:{{ school.Phone }}" class="btn
btn-outline-primary">
                            <i class="bi bi-telephone"></i> Позвонить в
школу
                        </a>

                        {% if school.Email %}
                        <a href="mailto:{{ school.Email }}" class="btn
btn-outline-success">
                            <i class="bi bi-envelope"></i> Написать
email
                        </a>
                        {% endif %}

                        <a href="{{ url_for('export_schools') }}"
class="btn btn-outline-info">
                            <i class="bi bi-download"></i> Экспорт
данных
                        </a>

                        <button class="btn btn-outline-secondary"
onclick="copyToClipboard('{{ school.Legal_Adress }}')">
                            <i class="bi bi-clipboard"></i> Копировать
адрес
                        </button>
                    </div>
                </div>
            </div>
        </div>
        <!-- Конец правой колонки -->
    </div>
</div>
{% endblock %}

{% block scripts %}
```

```
<script>
    function copyToClipboard(text) {
        navigator.clipboard.writeText(text).then(function() {
            alert('Адрес скопирован в буфер обмена');
        });
    }

    // Загрузка данных школы через API
    fetch('/api/school/{{ school.PK_School }}')
        .then(response => response.json())
        .then(data => {
            console.log('Данные школы:', data);
        })
        .catch(error => {
            console.error('Ошибка загрузки данных:', error);
        });
</script>
{% endblock %}
```

templates/base.html

```
<!DOCTYPE html>
<html lang="ru" data-bs-theme="light">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>{% block title %}Реестр школ Алтайского края{% endblock
%}</title>

    <!-- Bootstrap 5 -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m
in.css" rel="stylesheet">
```

```
    <!-- Bootstrap Icons -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.0/font/bootstra
p-icons.css">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">

    {% block head %}{% endblock %}
</head>
<body>
    <!-- Навигация -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary
fixed-top shadow-sm">
        <div class="container">
            <a class="navbar-brand" href="{{ url_for('index') }}">
                <i class="bi bi-building"></i>
                <span class="ms-2">Реестр школ Алтайского края</span>
            </a>

            <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarMain">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarMain">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('index')
}}">
                            <i class="bi bi-house"></i> Главная
                        </a>
                    </li>
```

```
                    <li class="nav-item">
                        <a class="nav-link" href="{{
url_for('search_schools') }}">
                            <i class="bi bi-search"></i> Поиск
                        </a>
                    </li>

                    {% if current_user.is_authenticated %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{
url_for('reports_index') }}">
                            <i class="bi bi-file-text"></i> Отчеты
                        </a>
                    </li>
                    {% endif %}

                     <!-- ВСТАВЬТЕ ЗДЕСЬ КОД ПРО ПРОВЕРКИ -->
                    <!-- {% if current_user.is_authenticated and
user_has_role('school_admin') %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{
url_for('admin_inspections') }}">
                            <i class="bi bi-clipboard-check"></i>
Проверки
                        </a>
                    </li>
                    {% endif %} -->
                    <!-- КОНЕЦ ВСТАВКИ -->

                    {% if current_user.is_authenticated and
user_has_role('school_admin') %}
<li class="nav-item">
    <a class="nav-link" href="/admin">
        <i class="bi bi-speedometer2"></i> Панель управления
    </a>
```

```
</li>
{% endif %}
                </ul>

                <ul class="navbar-nav">
                    {% if current_user.is_authenticated %}
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#"
role="button" data-bs-toggle="dropdown">
                            <i class="bi bi-person-circle"></i>
                            <span class="ms-1">{{ current_user.username
}}</span>
                            <small class="text-light ms-2">({{
user_role_name() }})</small>
                        </a>
                        <ul class="dropdown-menu dropdown-menu-end">
                            <li><a class="dropdown-item" href="{{
url_for('profile') }}">
                                <i class="bi bi-person"></i> Профиль
                            </a></li>
                            <li><hr class="dropdown-divider"></li>
                            <li><a class="dropdown-item" href="{{
url_for('logout') }}">
                                <i class="bi bi-box-arrow-right"></i>
Выйти
                            </a></li>
                        </ul>
                    </li>
                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('login')
}}">
                            <i class="bi bi-box-arrow-in-right"></i>
Войти
                        </a>
```

```
                    </li>

                    <li class="nav-item">
                        <a class="nav-link" href="{{
url_for('register') }}">
                            <i class="bi bi-person-plus"></i>
Регистрация
                        </a>
                    </li>

                    <li class="nav-item">
                        <a class="btn btn-outline-light ms-2" href="{{
url_for('auth_gosuslugi') }}">
                            <i class="bi bi-passport"></i> Госуслуги
                        </a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>

    <!-- Основное содержимое -->
    <main class="container mt-5 pt-3">
        <!-- Уведомления -->
        {% with messages = get_flashed_messages(with_categories=true)
%}
            {% if messages %}
                {% for category, message in messages %}
                    <div class="alert alert-{{ 'danger' if category ==
'error' else category }} alert-dismissible fade show mt-3"
role="alert">
                        {{ message }}
                        <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>
```

```
                    </div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        {% block content %}{% endblock %}
    </main>

    <!-- Подвал -->
    <footer class="footer mt-auto py-4 bg-dark text-white">
        <div class="container">
            <div class="row">
                <div class="col-md-4">
                    <h5>Реестр школ Алтайского края</h5>
                    <p class="mb-0">Министерство образования и науки Алтайского края</p>
                    <p class="text-muted small">© {{ now.year }} Все права защищены</p>
                </div>

                <div class="col-md-4">
                    <h6>Контакты</h6>
                    <ul class="list-unstyled">
                        <li><i class="bi bi-telephone"></i> 8 (3852) 29-44-07</li>
                        <li><i class="bi bi-envelope"></i> obraz@alreg.ru</li>
                        <li><i class="bi bi-geo-alt"></i> г. Барнаул, пр. Ленина, 54</li>
                    </ul>
                </div>

                <div class="col-md-4">
                    <h6>Полезные ссылки</h6>
```

```
                    <ul class="list-unstyled">
                        <li><a href="https://www.altairegion22.ru"
class="text-white-50 text-decoration-none">
                            <i class="bi bi-link-45deg"></i>
Правительство Алтайского края
                        </a></li>
                        <li><a href="https://www.gosuslugi.ru"
class="text-white-50 text-decoration-none">
                            <i class="bi bi-link-45deg"></i> Портал
госуслуг
                        </a></li>
                        <li><a href="http://www.obrnadzor.gov.ru"
class="text-white-50 text-decoration-none">
                            <i class="bi bi-link-45deg"></i>
Рособрнадзор
                        </a></li>
                    </ul>
                </div>
            </div>

            <hr class="my-4 text-white-50">

            <div class="row">
                <div class="col-12 text-center">
                    <p class="small text-white-50 mb-0">
                        Версия 1.0.0 |
                        <a href="#" class="text-white-50
text-decoration-none">Пользовательское соглашение</a> |
                        <a href="#" class="text-white-50
text-decoration-none">Политика конфиденциальности</a>
                    </p>
                </div>
            </div>
        </div>
    </footer>
```

```
    <!-- Скрипты -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bun
dle.min.js"></script>

    <script>
        // Активация всплывающих подсказок
        var tooltipTriggerList =
[].slice.call(document.querySelectorAll('[data-bs-toggle="tooltip"]'))
        var tooltipList = tooltipTriggerList.map(function
(tooltipTriggerEl) {
            return new bootstrap.Tooltip(tooltipTriggerEl)
        });

        // Активация всплывающих окон
        var popoverTriggerList =
[].slice.call(document.querySelectorAll('[data-bs-toggle="popover"]'))
        var popoverList = popoverTriggerList.map(function
(popoverTriggerEl) {
            return new bootstrap.Popover(popoverTriggerEl)
        });

        // Подтверждение действий
        document.addEventListener('DOMContentLoaded', function() {
            var confirmLinks =
document.querySelectorAll('.confirm-action');
            confirmLinks.forEach(function(link) {
                link.addEventListener('click', function(e) {
                    if (!confirm('Вы уверены?')) {
                        e.preventDefault();
                    }
                });
            });
        });
```

```
    </script>


    {% block scripts %}{% endblock %}
</body>
</html>
```

templates/index.html

```
{% extends "base.html" %}


{% block title %}Главная - Реестр школ Алтайского края{% endblock %}


{% block content %}


<div class="row">
    <!-- Левая колонка - фильтры -->
    <div class="col-lg-3 mb-4">
        <div class="card">
            <div class="card-header bg-primary text-white">
                <h5 class="mb-0"><i class="bi bi-funnel"></i>
Фильтры</h5>
            </div>
            <div class="card-body">
                <form method="get" action="{{ url_for('index') }}"
id="filter-form">
                    <!-- Район и населенный пункт -->
                    <div class="mb-3">
                        <label class="form-label">Район</label>
                        <select name="district_id" id="district-select"
class="form-select">
                            <option value="">Все районы</option>
                            {% for district in districts %}
                            <option value="{{ district.PK_District }}"
                                {% if request.args.get('district_id')
== district.PK_District|string %}selected{% endif %}>
```

```
                                {{ district.Name }}
                            </option>
                            {% endfor %}
                        </select>
                    </div>

                    <div class="mb-3">
                        <label class="form-label">Населенный
пункт</label>
                        <select name="settlement_id"
id="settlement-select" class="form-select">
                            <option value="">Все населенные
пункты</option>
                            {% for settlement in settlements %}
                            <option value="{{ settlement.PK_Settlement
}}"
                                {% if request.args.get('settlement_id')
== settlement.PK_Settlement|string %}selected{% endif %}>
                                {{ settlement.Type }} {{
settlement.Name }}
                            </option>
                            {% endfor %}
                        </select>
                    </div>

                    <!-- Тип школы -->
                    <div class="mb-3">
                        <label class="form-label">Тип школы</label>
                        <select name="school_type_id"
class="form-select">
                            <option value="">Все типы</option>
                            {% for type in school_types %}
                            <option value="{{ type.PK_Type_of_School
}}"
```

```
                                {% if
request.args.get('school_type_id') == type.PK_Type_of_School|string
%}selected{% endif %}>

                                {{ type.Name }}

                            </option>

                            {% endfor %}

                        </select>

                    </div>


                    <!-- Специализации (профильные классы) -->

                    <div class="mb-3">

                        <label class="form-label">Специализации</label>

                        <select name="specialization_id"
class="form-select">

                            <option value="">Любая
специализация</option>

                            {% for spec in specializations %}

                            <option value="{{ spec.PK_Specialization
}}"

                                {% if
request.args.get('specialization_id') == spec.PK_Specialization|string
%}selected{% endif %}>

                                {{ spec.Name }}

                            </option>

                            {% endfor %}

                        </select>

                    </div>


                    <!-- Инфраструктура -->

                    <div class="mb-3">

                        <label
class="form-label">Инфраструктура</label>

                        <div class="form-check">

                            <input class="form-check-input"
type="checkbox" name="has_library" id="has_library"
```

```
                                value="1" {% if
request.args.get('has_library') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_library">
                                Библиотека
                            </label>
                        </div>
                        <div class="form-check">
                            <input class="form-check-input"
type="checkbox" name="has_gym" id="has_gym"
                                value="1" {% if
request.args.get('has_gym') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_gym">
                                Спортзал
                            </label>
                        </div>
                        <div class="form-check">
                            <input class="form-check-input"
type="checkbox" name="has_lab" id="has_lab"
                                value="1" {% if
request.args.get('has_lab') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_lab">
                                Лаборатория
                            </label>
                        </div>
                        <div class="form-check">
                            <input class="form-check-input"
type="checkbox" name="has_accessible" id="has_accessible"
                                value="1" {% if
request.args.get('has_accessible') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_accessible">
                                Доступная среда для лиц с ОВЗ
                            </label>
```

```
                    </div>
                </div>

                <!-- Количество учащихся -->
                <div class="mb-3">
                    <label class="form-label">Количество
учащихся</label>
                    <div class="input-group mb-2">
                        <span class="input-group-text">От</span>
                        <input type="number" name="min_students"
class="form-control"
                               value="{{
request.args.get('min_students', '') }}"
                               placeholder="0" min="0">
                    </div>
                    <div class="input-group">
                        <span class="input-group-text">До</span>
                        <input type="number" name="max_students"
class="form-control"
                               value="{{
request.args.get('max_students', '') }}"
                               placeholder="5000" min="0">
                    </div>
                </div>

                <!-- Дополнительные программы образования -->
                <div class="mb-3">
                    <label class="form-label">Дополнительные
программы</label>
                    <select name="program_type"
class="form-select">
                        <option value="">Любые программы</option>
                        <option value="дополнительная" {% if
request.args.get('program_type') == 'дополнительная' %}selected{% endif
%}>
                            Дополнительные программы
```

```
                            </option>
                            <option value="основная" {% if
request.args.get('program_type') == 'основная' %}selected{% endif %}>
                                Основные программы
                            </option>
                        </select>
                    </div>

                    <!-- Год основания -->
                    <div class="mb-3">
                        <label class="form-label">Год основания</label>
                        <div class="input-group mb-2">
                            <span class="input-group-text">От</span>
                            <input type="number" name="min_year"
class="form-control"
                                value="{{ request.args.get('min_year',
'1900') }}"
                                placeholder="1900" min="1900"
max="2026">
                        </div>
                        <div class="input-group">
                            <span class="input-group-text">До</span>
                            <input type="number" name="max_year"
class="form-control"
                                value="{{ request.args.get('max_year',
'2026') }}"
                                placeholder="2026" min="1900"
max="2026">
                        </div>
                    </div>

                    <!-- Аккредитация и лицензия -->
                    <div class="mb-3">
                        <div class="form-check">
```

```
                            <input class="form-check-input"
type="checkbox" name="has_accreditation" id="has_accreditation"
                                value="1" {% if
request.args.get('has_accreditation') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_accreditation">
                                Имеет аккредитацию
                            </label>
                        </div>
                        <div class="form-check">
                            <input class="form-check-input"
type="checkbox" name="has_license" id="has_license"
                                value="1" {% if
request.args.get('has_license') %}checked{% endif %}>
                            <label class="form-check-label"
for="has_license">
                                Имеет лицензию
                            </label>
                        </div>
                    </div>

                    <div class="d-grid gap-2">
                        <button type="submit" class="btn btn-primary">
                            <i class="bi bi-funnel-fill"></i> Применить
фильтры
                        </button>
                        <a href="{{ url_for('index') }}" class="btn
btn-outline-secondary">
                            <i class="bi bi-x-circle"></i> Сбросить все
фильтры
                        </a>
                    </div>
                </form>
            </div>
        </div>
```

```
        <!-- Быстрая статистика -->
        <div class="card mt-4">
            <div class="card-header bg-info text-white">
                <h6 class="mb-0"><i class="bi bi-bar-chart"></i>
Статистика</h6>
            </div>
            <div class="card-body">
                <div class="d-flex justify-content-between mb-2">
                    <span>Всего школ:</span>
                    <strong>{{ schools.total }}</strong>
                </div>
                <div class="d-flex justify-content-between mb-2">
                    <span>На странице:</span>
                    <strong>{{ schools.items|length }}</strong>
                </div>
                {% if current_user.is_authenticated and
user_has_role('region_admin') %}
                <hr>
                <a href="{{ url_for('export_schools') }}" class="btn
btn-sm btn-outline-success w-100">
                    <i class="bi bi-download"></i> Экспорт всех школ
                </a>
                {% endif %}
            </div>
        </div>
    </div>

    <!-- Правая колонка - контент -->
    <div class="col-lg-9">
        <!-- Заголовок и поиск -->
        <div class="d-flex justify-content-between align-items-center
mb-4">
            <div>
```

```
                <h1 class="h3 mb-0">Образовательные учреждения
Алтайского края</h1>

                <p class="text-muted mb-0">Найдено {{ schools.total }}
школ</p>

                <!-- Активные фильтры -->

                {% if has_active_filters %}

                <div class="mt-2">

                    <small class="text-muted">Активные фильтры:</small>

                    {% for key, value in active_filters %}

                    <span class="badge bg-info me-1">

                        {{ key }}: {{ value }}

                        <a href="{{ remove_filter_url(key) }}"
class="text-white ms-1">

                            <i class="bi bi-x"></i>

                        </a>

                    </span>

                    {% endfor %}

                </div>

                {% endif %}

            </div>

            {% if current_user.is_authenticated and
user_has_role('region_admin') %}

            <a href="{{ url_for('add_school') }}" class="btn
btn-success">

                <i class="bi bi-plus-circle"></i> Добавить школу

            </a>

            {% endif %}

        </div>



        <!-- Поисковая строка -->

        <div class="card mb-4">

            <div class="card-body">

                <form method="get" action="{{ url_for('search_schools')
}}" class="row g-2">

                    <div class="col-md-10">
```

```
                        <div class="input-group">
                            <span class="input-group-text">
                                <i class="bi bi-search"></i>
                            </span>
                            <input type="text" name="q"
class="form-control"
                                placeholder="Поиск по названию школы,
адресу или телефону..."
                                value="{{ request.args.get('q', '')
}}">
                        </div>
                    </div>
                    <div class="col-md-2">
                        <button type="submit" class="btn btn-primary
w-100">
                            Найти
                        </button>
                    </div>
                </form>
            </div>
        </div>


        <!-- Сортировка -->
<div class="d-flex justify-content-between align-items-center mb-3">
    {% set args_without_sort = {} %}
    {% for key, value in request.args.items() %}
        {% if key not in ['sort_by', 'order'] %}
            {% set _ = args_without_sort.update({key: value}) %}
        {% endif %}
    {% endfor %}

    <div class="btn-group" role="group">
        <a href="{{ url_for('index', sort_by='name', order='asc',
**args_without_sort) }}"
```

```
            class="btn btn-outline-secondary {% if
request.args.get('sort_by', 'name') == 'name' and
request.args.get('sort_by') != 'students' and
request.args.get('sort_by') != 'rating' %}active{% endif %}">

            По названию

        </a>

        <a href="{{ url_for('index', sort_by='students', order='desc',
**args_without_sort) }}"

            class="btn btn-outline-secondary {% if
request.args.get('sort_by') == 'students' %}active{% endif %}">

            По количеству учащихся

        </a>

        <a href="{{ url_for('index', sort_by='rating', order='desc',
**args_without_sort) }}"

            class="btn btn-outline-secondary {% if
request.args.get('sort_by') == 'rating' %}active{% endif %}">

            По рейтингу

        </a>

    </div>


    <div class="text-muted small">

        Страница {{ schools.page }} из {{ schools.pages }}

    </div>

</div>


        <!-- Список школ -->

        {% if schools.items %}

        <div class="row row-cols-1 row-cols-md-2 g-4">

            {% for school in schools.items %}

            <div class="col">

                <div class="card h-100 shadow-sm school-card">

                    <div class="card-body">

                        <div class="d-flex justify-content-between
align-items-start mb-2">

                            <h5 class="card-title mb-0">{{
school.Official_Name }}</h5>
```

```
                            {% if not school.is_active %}
                            <span class="badge
bg-danger">Неактивна</span>
                            {% endif %}
                        </div>

                        <h6 class="card-subtitle mb-3 text-muted">
                            <i class="bi bi-building"></i>
                            {{ school.type_of_school.Name if
school.type_of_school else 'Тип не указан' }}
                        </h6>

                        <p class="card-text">
                            <i class="bi bi-geo-alt"></i>
                            {{ school.Legal_Adress[:80] }}{% if
school.Legal_Adress|length > 80 %}...{% endif %}
                        </p>

                        <div class="mb-3">
                            <small class="text-muted">
                                <i class="bi bi-telephone"></i> {{
school.Phone }}
                            </small>
                        </div>

                        {% if school.Number_of_Students %}
                        <div class="mb-3">
                            <span class="badge bg-info">
                                <i class="bi bi-people"></i> {{
school.Number_of_Students }} учащихся
                            </span>
                        </div>
                        {% endif %}
```

```
                        <!-- Инфраструктура и специализации (кратко)
-->

                        <div class="mb-3">

                            {% if school.infrastructure %}

                            <div class="d-flex flex-wrap gap-1">

                                {% for infra in
school.infrastructure[:3] %}

                                <span class="badge bg-light text-dark">

                                    <i class="bi bi-check-circle"></i>
{{ infra.Name }}

                                </span>

                                {% endfor %}

                                {% if school.infrastructure|length > 3
%}

                                <span class="badge bg-light
text-dark">+{{ school.infrastructure|length - 3 }}</span>

                                {% endif %}

                            </div>

                            {% endif %}

                        </div>



                        <!-- Рейтинг -->

                        <div class="mb-3">

                            {% set avg_rating = school.get_avg_rating()
%}

                            {% set review_count =
school.get_review_count() %}

                            {% if review_count > 0 %}

                            <div class="d-flex align-items-center">

                                {{ render_star_rating(avg_rating)|safe
}}

                                <span class="ms-2">{{
avg_rating|round(1) }}</span>

                                <small class="text-muted ms-2">({{
review_count }} отзывов)</small>

                            </div>
```

```
                            {% else %}
                            <small class="text-muted">Нет
отзывов</small>
                            {% endif %}
                        </div>

                        <div class="d-flex justify-content-between">
                            <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}"
                               class="btn btn-outline-primary btn-sm">
                                <i class="bi bi-info-circle"></i>
Подробнее
                            </a>
                            {% if current_user.is_authenticated and
user_has_role('school_admin') %}
                            <a href="{{ url_for('edit_school',
school_id=school.PK_School) }}"
                               class="btn btn-outline-warning btn-sm">
                                <i class="bi bi-pencil"></i>
Редактировать
                            </a>
                            {% endif %}
                        </div>
                    </div>

                    <div class="card-footer bg-transparent">
                        <div class="d-flex justify-content-between
align-items-center">
                            <small class="text-muted">
                                <i class="bi bi-calendar"></i>
                                {% if school.Founding_Date %}
                                Основана в {{ school.Founding_Date.year
}} году
                                {% else %}
                                Дата основания не указана
```

```
                            {% endif %}
                        </small>
                        <small class="text-muted">
                            ID: {{ school.PK_School }}
                        </small>
                    </div>
                </div>
            </div>
        </div>
        {% endfor %}
    </div>

    <!-- Пагинация -->
    {% if schools.pages > 1 %}
    <nav aria-label="Навигация по страницам" class="mt-4">
        <ul class="pagination justify-content-center">
            {% set args_without_page = {} %}
            {% for key, value in request.args.items() %}
            {% if key != 'page' %}
            {% set _ = args_without_page.update({key: value}) %}
            {% endif %}
            {% endfor %}

            <!-- Первая страница -->
            {% if schools.page > 1 %}
            <li class="page-item">
                <a class="page-link" href="{{ url_for('index',
page=1, **args_without_page) }}">
                    <i class="bi bi-chevron-double-left"></i>
                </a>
            </li>
            {% endif %}
```

```
                <!-- Предыдущая страница -->
                {% if schools.has_prev %}
                <li class="page-item">
                    <a class="page-link" href="{{ url_for('index',
page=schools.prev_num, **args_without_page) }}">
                        <i class="bi bi-chevron-left"></i>
                    </a>
                </li>
                {% endif %}

                <!-- Номера страниц -->
                {% for page_num in schools.iter_pages(left_edge=1,
left_current=2, right_current=3, right_edge=1) %}
                {% if page_num %}
                <li class="page-item {% if page_num == schools.page
%}active{% endif %}">
                    <a class="page-link" href="{{ url_for('index',
page=page_num, **args_without_page) }}">
                        {{ page_num }}
                    </a>
                </li>
                {% else %}
                <li class="page-item disabled">
                    <span class="page-link">...</span>
                </li>
                {% endif %}
                {% endfor %}

                <!-- Следующая страница -->
                {% if schools.has_next %}
                <li class="page-item">
                    <a class="page-link" href="{{ url_for('index',
page=schools.next_num, **args_without_page) }}">
                        <i class="bi bi-chevron-right"></i>
```

```
                    </a>
                </li>
                {% endif %}

                <!-- Последняя страница -->
                {% if schools.page < schools.pages %}
                <li class="page-item">
                    <a class="page-link" href="{{ url_for('index',
page=schools.pages, **args_without_page) }}">
                        <i class="bi bi-chevron-double-right"></i>
                    </a>
                </li>
                {% endif %}
            </ul>

            <div class="text-center mt-2">
                <small class="text-muted">
                    Показано {{ ((schools.page - 1) * schools.per_page)
+ 1 }} -
                    {{ [schools.page * schools.per_page,
schools.total]|min }} из {{ schools.total }} школ
                </small>
            </div>
        </nav>
        {% endif %}

        {% else %}
        <!-- Нет школ -->
        <div class="text-center py-5">
            <div class="mb-4">
                <i class="bi bi-search display-1 text-muted"></i>
            </div>
            <h3 class="h4 mb-3">Школы не найдены</h3>
```

```
            <p class="text-muted mb-4">
                Попробуйте изменить критерии поиска или фильтры
            </p>
            <a href="{{ url_for('index') }}" class="btn btn-primary">
                <i class="bi bi-arrow-clockwise"></i> Сбросить фильтры
            </a>
        </div>
        {% endif %}
    </div>
</div>

<script>
document.addEventListener('DOMContentLoaded', function() {
    // Динамическая загрузка населенных пунктов при выборе района
    const districtSelect = document.getElementById('district-select');
    const settlementSelect =
document.getElementById('settlement-select');

    if (districtSelect) {
        districtSelect.addEventListener('change', function() {
            const districtId = this.value;

            if (districtId) {
                fetch(`/api/districts/${districtId}/settlements`)
                    .then(response => response.json())
                    .then(data => {
                        settlementSelect.innerHTML = '<option
value="">Все населенные пункты</option>';
                        data.forEach(function(settlement) {
                            const option =
document.createElement('option');
                            option.value = settlement.id;
                            option.textContent = settlement.type + ' '
+ settlement.name;
```

```
                        settlementSelect.appendChild(option);
                    });

                    // Сохраняем выбранный населенный пункт если он
есть
                    const currentSettlement = "{{
request.args.get('settlement_id', '') }}";
                    if (currentSettlement) {
                        settlementSelect.value = currentSettlement;
                    }
                })
                .catch(error => {
                    console.error('Ошибка при загрузке населенных
пунктов:', error);
                });
        } else {
            // Загружаем все населенные пункты если район не выбран
            fetch('/api/settlements')
                .then(response => response.json())
                .then(data => {
                    settlementSelect.innerHTML = '<option
value="">Все населенные пункты</option>';
                    data.forEach(function(settlement) {
                        const option =
document.createElement('option');
                        option.value = settlement.id;
                        option.textContent = settlement.type + ' '
+ settlement.name;
                        settlementSelect.appendChild(option);
                    });
                });
        }
    });

    // Инициализация при загрузке страницы
```

```
        if (districtSelect.value) {
            districtSelect.dispatchEvent(new Event('change'));
        }
    }
});
</script>

{% endblock %}
```

templates/search_results.html

```
{% extends "base.html" %}

{% block title %}Результаты поиска - Реестр школ Алтайского края{% endblock %}

{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-4">
        <div>
            <h1 class="h3 mb-0">Результаты поиска</h1>
            <p class="text-muted mb-0">
                {% if schools.total %}
                Найдено {{ schools.total }} школ по запросу "{{ query }}"
                {% else %}
                По запросу "{{ query }}" ничего не найдено
                {% endif %}
            </p>
        </div>
        <a href="{{ url_for('index') }}" class="btn btn-outline-secondary">
            <i class="bi bi-arrow-left"></i> Вернуться к списку
        </a>
```

```
    </div>


    {% if schools.total %}

    <div class="row row-cols-1 row-cols-md-2 g-4">

        {% for school in schools.items %}

        <div class="col">

            <div class="card h-100 shadow-sm school-card">

                <div class="card-body">

                    <h5 class="card-title mb-2">{{ school.Official_Name
}}</h5>

                    <h6 class="card-subtitle mb-3 text-muted">

                        <i class="bi bi-building"></i>

                        {{ school.type_of_school.Name if
school.type_of_school else 'Тип не указан' }}

                    </h6>

                    <p class="card-text">

                        <i class="bi bi-geo-alt"></i>

                        {{ school.Legal_Adress[:80] }}{% if
school.Legal_Adress|length > 80 %}...{% endif %}

                    </p>

                    <div class="mb-3">

                        <small class="text-muted">

                            <i class="bi bi-telephone"></i> {{
school.Phone }}

                        </small>

                    </div>

                    {% if school.Number_of_Students %}

                    <div class="mb-3">

                        <span class="badge bg-info">

                            <i class="bi bi-people"></i> {{
school.Number_of_Students }} учащихся

                        </span>

                    </div>

                    {% endif %}
```

```
                    <div class="d-flex justify-content-between">
                        <a href="{{ url_for('school_detail',
school_id=school.PK_School) }}"
                           class="btn btn-outline-primary btn-sm">
                            <i class="bi bi-info-circle"></i> Подробнее
                        </a>
                    </div>
                </div>
            </div>
        </div>
        {% endfor %}
    </div>

    {% if schools.pages > 1 %}
    <nav aria-label="Навигация по страницам" class="mt-4">
        <ul class="pagination justify-content-center">
            {% if schools.has_prev %}
            <li class="page-item">
                <a class="page-link" href="{{ url_for('search_schools',
page=schools.prev_num, q=query) }}">
                    <i class="bi bi-chevron-left"></i>
                </a>
            </li>
            {% endif %}

            {% for page_num in schools.iter_pages(left_edge=1,
right_edge=1, left_current=2, right_current=2) %}
            {% if page_num %}
            <li class="page-item {% if page_num == schools.page
%}active{% endif %}">
                <a class="page-link" href="{{ url_for('search_schools',
page=page_num, q=query) }}">
                    {{ page_num }}
                </a>
```

```
            </li>
            {% else %}
            <li class="page-item disabled">
                <span class="page-link">...</span>
            </li>
            {% endif %}
            {% endfor %}

            {% if schools.has_next %}
            <li class="page-item">
                <a class="page-link" href="{{ url_for('search_schools',
page=schools.next_num, q=query) }}">
                    <i class="bi bi-chevron-right"></i>
                </a>
            </li>
            {% endif %}
        </ul>
    </nav>
    {% endif %}

    {% else %}
    <div class="text-center py-5">
        <div class="mb-4">
            <i class="bi bi-search display-1 text-muted"></i>
        </div>
        <h3 class="h4 mb-3">Ничего не найдено</h3>
        <p class="text-muted mb-4">
            Попробуйте изменить критерии поиска
        </p>
        <a href="{{ url_for('index') }}" class="btn btn-primary">
            <i class="bi bi-arrow-clockwise"></i> Вернуться к списку
школ
        </a>
```

```
    </div>

    {% endif %}

</div>

{% endblock %}
```

.env

```
SECRET_KEY=ваш-секретный-ключ-минимум-32-символа-изменять-в-production

DATABASE_URL=postgresql://postgres:postgres@localhost:5432/school_regis
try

FLASK_APP=app.py

FLASK_ENV=development

DEBUG=True

GOSUSLUGI_APP_ID=demo-app-id

GOSUSLUGI_REDIRECT_URI=http://localhost:5000/auth/gosuslugi/callback
```

app.py import os

import csv

import json

from datetime import datetime, timezone

from functools import wraps

import subprocess

from datetime import datetime

from flask import (

    Flask, render_template, request, jsonify,

    redirect, url_for, flash, send_file, abort,

    make_response, session, send_from_directory

)

from flask_login import (

    LoginManager, login_user, logout_user,

    login_required, current_user

```
)
from flask_migrate import Migrate
from config import config
from models import db, User, School, District, Settlement, TypeOfSchool,
Infrastructure
from models import Specialization, Employee, Subject, EducationProgram,
Review, Inspection
from models import AuditLog, ImportHistory
from forms import (
    LoginForm, RegistrationForm, SchoolForm,
    ReviewForm, ImportForm, EmployeeForm,
    ProgramForm, UserProfileForm
)
from models import AuditLog, ImportHistory, SchoolVersion
from utils import (
    allowed_file, audit_log, role_required,
    export_to_csv, export_to_excel, import_from_csv, import_from_excel,
    validate_school_data, generate_report_stats, create_backup,
    render_star_rating, pluralize,
    save_school_version_on_create,
    save_school_version_on_update,
    save_school_version_on_delete,
    get_school_versions,  create_version
)
from sqlalchemy import func
```

```
# Инициализация приложения
app = Flask(__name__)

# Загрузка конфигурации
app.config.from_object(config['development'])
config['development'].init_app(app)

# Инициализация расширений
db.init_app(app)
migrate = Migrate(app, db)

login_manager = LoginManager(app)
login_manager.login_view = 'login'
login_manager.login_message = 'Пожалуйста, войдите для доступа к этой странице.'
login_manager.login_message_category = 'warning'

@login_manager.user_loader
def load_user(user_id):
    return db.session.get(User, int(user_id))
def get_user_by_id(user_id):
    """Получить пользователя по ID"""
    from models import User
    return User.query.get(user_id)
```

```
def get_school_type_by_id(type_id):
    """Получить тип школы по ID"""
    from models import TypeOfSchool
    return TypeOfSchool.query.get(type_id)


def get_settlement_by_id(settlement_id):
    """Получить населенный пункт по ID"""
    from models import Settlement
    return Settlement.query.get(settlement_id)


# Контекстные процессоры
# Контекстные процессоры
# Контекстные процессоры
# Контекстные процессоры
@app.context_processor
def inject_globals():
    from config import Config
    import json


    def user_has_role(role_name):
        """Безопасная проверка роли пользователя в шаблонах"""
        if not current_user.is_authenticated:
            return False


        from models import User
```

```
    user = User.query.get(current_user.id)

    if not user:
        return False

    # Маппинг имен ролей на числовые значения
    role_mapping = {
        'parent_student': 1,
        'school_employee': 2,
        'other': 3,
        'admin': 4,
        'super_admin': 5
    }

    required_role = role_mapping.get(role_name, 0)
    return user.role >= required_role

def can_edit_school_wrapper(school_id):
    """Обертка для can_edit_school для использования в шаблонах"""
    return can_edit_school(school_id)

def user_role_name():
    """Безопасное получение имени роли в шаблонах"""
    if not current_user.is_authenticated:
        return 'Гость'
```

```
    from models import User
    user = User.query.get(current_user.id)

    if not user:
        return 'Неизвестно'

    roles = {
        1: 'Ученик/Родитель',
        2: 'Работник учреждения',
        3: 'Другой пользователь',
        4: 'Администратор',
        5: 'Супер-администратор'
    }
    return roles.get(user.role, 'Неизвестно')

def get_filter_display_name(key, value):
    """Получить читаемое имя для фильтра"""
    if key == 'district_id':
        district = District.query.get(value)
        return f"Район: {district.Name}" if district else value
    elif key == 'settlement_id':
        settlement = Settlement.query.get(value)
        return f"Нас. пункт: {settlement.Name}" if settlement else value
    elif key == 'school_type_id':
```

```
            school_type = TypeOfSchool.query.get(value)
            return f"Тип: {school_type.Name}" if school_type else value
        elif key == 'specialization_id':
            spec = Specialization.query.get(value)
            return f"Специализация: {spec.Name}" if spec else value
        elif key == 'has_library':
            return "Библиотека"
        elif key == 'has_gym':
            return "Спортзал"
        elif key == 'has_lab':
            return "Лаборатория"
        elif key == 'has_accessible':
            return "Доступная среда"
        elif key == 'program_type':
            return "Программа: " + ("Дополнительная" if value ==
"дополнительная" else "Основная")
        elif key == 'min_students':
            return f"Учащихся от: {value}"
        elif key == 'max_students':
            return f"Учащихся до: {value}"
        elif key == 'min_year':
            return f"Год от: {value}"
        elif key == 'max_year':
            return f"Год до: {value}"
        return f"{key}: {value}"
```

```
def get_active_filters():
    """Получить список активных фильтров для отображения"""
    filters = []
    for key, value in request.args.items():
        if key not in ['page', 'sort_by', 'order', 'q'] and value:
            if isinstance(value, list):
                for v in value:
                    filters.append((key, get_filter_display_name(key, v)))
            else:
                filters.append((key, get_filter_display_name(key, value)))
    return filters


def has_active_filters():
    """Проверить, есть ли активные фильтры"""
    for key in request.args:
        if key not in ['page', 'sort_by', 'order', 'q'] and request.args[key]:
            return True
    return False


def remove_filter_url(filter_key):
    """Создать URL без указанного фильтра"""
    args = dict(request.args)
    if 'page' in args:
        del args['page']
```

```
    if filter_key in args:
        del args[filter_key]
    return url_for('index', **args)


def fromjson(value):
    """Фильтр Jinja2 для парсинга JSON"""
    if not value:
        return None
    try:
        if isinstance(value, str):
            # Убедимся, что правильно парсим Unicode
            return json.loads(value)
        elif hasattr(value, '__dict__'):
            # Если это уже объект
            return value
        return value
    except Exception as e:
        print(f"Ошибка парсинга JSON: {e}")
        return None


def get_field_display_name(field):
    """Получить читаемое название поля"""
    field_names = {
        'Official_Name': 'Официальное название',
        'Legal_Adress': 'Юридический адрес',
```

```
        'Phone': 'Телефон',
        'Email': 'Email',
        'Website': 'Веб-сайт',
        'Founding_Date': 'Дата основания',
        'Number_of_Students': 'Количество учащихся',
        'License': 'Лицензия',
        'Accreditation': 'Аккредитация',
        'PK_Type_of_School': 'Тип школы',
        'PK_Settlement': 'Населенный пункт',
        'is_active': 'Статус активности'
    }
    return field_names.get(field, field)

def safe_json_display(value):
    """Безопасное отображение JSON значения"""
    if value is None:
        return '(пусто)'
    if isinstance(value, str):
        return value
    return str(value)

return {
    'roles': Config.ROLES,
    'current_user': current_user,
    'now': datetime.now(),
```

```
        'render_star_rating': render_star_rating,
        'pluralize': pluralize,
        'user_has_role': user_has_role,
        'user_role_name': user_role_name,
        'can_edit_school': can_edit_school_wrapper,
        'get_filter_display_name': get_filter_display_name,
        'get_active_filters': get_active_filters,
        'has_active_filters': has_active_filters(),
        'active_filters': get_active_filters(),
        'remove_filter_url': remove_filter_url,
        'fromjson': fromjson,
        'get_field_display_name': get_field_display_name,
        'safe_json_display': safe_json_display,
        'get_user_by_id': get_user_by_id,
        'get_school_type_by_id': get_school_type_by_id,
        'get_settlement_by_id': get_settlement_by_id
    }


@app.template_filter('fromjson')
def fromjson_filter(value):
    """Фильтр Jinja2 для парсинга JSON в шаблонах."""
    if not value:
        return None
    try:
        if isinstance(value, str):
```

```
            return json.loads(value)
        elif hasattr(value, '__dict__'):
            return value
        return value
    except Exception as e:
        print(f"Ошибка парсинга JSON: {e}")
        return None

# Обработка ошибок
@app.errorhandler(404)
def not_found_error(error):
    return render_template('errors/404.html'), 404

@app.errorhandler(403)
def forbidden_error(error):
    return render_template('errors/403.html'), 403

@app.errorhandler(500)
def internal_error(error):
    db.session.rollback()
    return render_template('errors/500.html'), 500

# Статические файлы
@app.route('/static/<path:filename>')
def static_files(filename):
```

```
    return send_from_directory('static', filename)


# Главная страница

# Главная страница

# Главная страница

@app.route('/')

def index():

    page = request.args.get('page', 1, type=int)

    per_page = app.config['SCHOOLS_PER_PAGE']


    # Фильтрация

    query = School.query.filter_by(is_active=True)


    # Применение фильтров

    district_id = request.args.get('district_id')

    settlement_id = request.args.get('settlement_id')

    school_type_id = request.args.get('school_type_id')

    specialization_id = request.args.get('specialization_id')

    min_students = request.args.get('min_students')

    max_students = request.args.get('max_students')

    min_year = request.args.get('min_year')

    max_year = request.args.get('max_year')

    program_type = request.args.get('program_type')


    # Инфраструктура
```

```
    has_library = request.args.get('has_library')
    has_gym = request.args.get('has_gym')
    has_lab = request.args.get('has_lab')
    has_accessible = request.args.get('has_accessible')
    has_accreditation = request.args.get('has_accreditation')
    has_license = request.args.get('has_license')

    if district_id:
        query = query.join(Settlement).filter(Settlement.PK_District == district_id)

    if settlement_id:
        query = query.filter(School.PK_Settlement == settlement_id)

    if school_type_id:
        query = query.filter(School.PK_Type_of_School == school_type_id)

    if specialization_id:
        query =
query.filter(School.specializations.any(PK_Specialization=specialization_id))

    if min_students:
        query = query.filter(School.Number_of_Students >= min_students)

    if max_students:
        query = query.filter(School.Number_of_Students <= max_students)
```

```
    if min_year:
        query = query.filter(db.extract('year', School.Founding_Date) >=
min_year)

    if max_year:
        query = query.filter(db.extract('year', School.Founding_Date) <=
max_year)

    # Фильтры по инфраструктуре
    if has_library:
        library = Infrastructure.query.filter_by(Name='Библиотека').first()
        if library:
            query =
query.filter(School.infrastructure.any(PK_Infrastructure=library.PK_Infrastruct
ure))

    if has_gym:
        gym = Infrastructure.query.filter_by(Name='Спортзал').first()
        if gym:
            query =
query.filter(School.infrastructure.any(PK_Infrastructure=gym.PK_Infrastructure
))

    if has_lab:
        lab = Infrastructure.query.filter_by(Name='Лаборатория').first()
```

```
        if lab:
            query = query.filter(School.infrastructure.any(PK_Infrastructure=lab.PK_Infrastructure))

    if has_accessible:
        accessible = Infrastructure.query.filter(Infrastructure.Name.ilike('%доступн%')).first()
        if accessible:
            query = query.filter(School.infrastructure.any(PK_Infrastructure=accessible.PK_Infrastructure))

    # Фильтры по аккредитации и лицензии
    if has_accreditation:
        query = query.filter(School.Accreditation != None, School.Accreditation != '')

    if has_license:
        query = query.filter(School.License != None, School.License != '')

    # Фильтр по программам образования
    if program_type:
        query = query.filter(School.programs.any(Type=program_type))

    # Сортировка
```

```
    sort_by = request.args.get('sort_by', 'name')
    order = request.args.get('order', 'asc')

    if sort_by == 'students':
        if order == 'desc':
            query = query.order_by(db.desc(School.Number_of_Students))
        else:
            query = query.order_by(School.Number_of_Students)
    elif sort_by == 'rating':
        # Создаем подзапрос для рейтинга с количеством отзывов
        subquery = db.session.query(
            Review.PK_School,
            db.func.coalesce(db.func.avg(Review.Rating), 0).label('avg_rating'),
            db.func.count(Review.PK_Review).label('review_count')
        ).filter(
            Review.is_approved == True
        ).group_by(
            Review.PK_School
        ).subquery()

        # Делаем LEFT JOIN с подзапросом
        query = query.outerjoin(subquery, School.PK_School ==
subquery.c.PK_School)
```

```
        # Для сортировки используем CASE, чтобы школы без отзывов имели рейтинг 0
        # и шли в конце при сортировке по убыванию
        if order == 'desc':
            # Сначала школы с рейтингом от 5 до 1, затем с рейтингом 0 (без отзывов)
            query = query.order_by(
                db.desc(db.case(
                    (subquery.c.review_count > 0, subquery.c.avg_rating),
                    else_=0
                ))
            )
        else:
            # Сначала школы с рейтингом от 0 до 5
            query = query.order_by(
                db.case(
                    (subquery.c.review_count > 0, subquery.c.avg_rating),
                    else_=0
                ).asc()
            )
    else:  # Сортировка по названию по умолчанию
        if order == 'desc':
            query = query.order_by(db.desc(School.Official_Name))
        else:
            query = query.order_by(School.Official_Name)
```

```
    # Пагинация
    try:
        schools = query.paginate(page=page, per_page=per_page, error_out=False)
    except:
        schools = query.paginate(page=1, per_page=per_page, error_out=False)

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    settlements = Settlement.query.order_by(Settlement.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()
    specializations = Specialization.query.order_by(Specialization.Name).all()
    infrastructure_items =
Infrastructure.query.order_by(Infrastructure.Name).all()

    return render_template('index.html',
        schools=schools,
        districts=districts,
        settlements=settlements,
        school_types=school_types,
        specializations=specializations,
        infrastructure_items=infrastructure_items
    )

# API для получения населенных пунктов по району
```

```
@app.route('/api/districts/<int:district_id>/settlements')
def get_settlements_by_district(district_id):
    settlements = Settlement.query.filter_by(PK_District=district_id).all()
    return jsonify([{
        'id': s.PK_Settlement,
        'name': s.Name,
        'type': s.Type
    } for s in settlements])

# Поиск школ
@app.route('/search')
def search_schools():
    query = request.args.get('q', '').strip()
    page = request.args.get('page', 1, type=int)

    if not query:
        return redirect(url_for('index'))

    search_results = School.query.filter(
        db.or_(
            School.Official_Name.ilike(f'%{query}%'),
            School.Legal_Adress.ilike(f'%{query}%'),
            School.Phone.ilike(f'%{query}%')
        )
    ).filter_by(is_active=True).paginate(page=page, per_page=20)
```

```
    return render_template('search_results.html',
        schools=search_results,
        query=query)

# Детальная информация о школе
@app.route('/school/<int:school_id>')
def school_detail(school_id):
    school = School.query.get_or_404(school_id)

    if not school.is_active and not current_user.has_role('school_admin'):
        abort(404)

    # Статистика отзывов
    reviews = Review.query.filter_by(
        PK_School=school_id,
        is_approved=True
    ).order_by(db.desc(Review.Date)).limit(10).all()

    avg_rating = school.get_avg_rating()
    review_count = school.get_review_count()

    # Инспекции
    inspections = Inspection.query.filter_by(
        PK_School=school_id
```

```
    ).order_by(db.desc(Inspection.Date)).limit(5).all()

    # Сотрудники
    employees = school.employees[:10] if school.employees else []

    # Программы
    programs = school.programs[:10] if school.programs else []

    form = ReviewForm()

    return render_template('schools/detail.html',
        school=school,
        reviews=reviews,
        avg_rating=avg_rating,
        review_count=review_count,
        inspections=inspections,
        employees=employees,
        programs=programs,
        form=form)

# API для школы
@app.route('/api/school/<int:school_id>')
def api_school(school_id):
    school = School.query.get_or_404(school_id)
```

```
    if not school.is_active and not current_user.has_role('school_admin'):
        abort(404)

    data = {
        'id': school.PK_School,
        'name': school.Official_Name,
        'address': school.Legal_Adress,
        'phone': school.Phone,
        'email': school.Email,
        'website': school.Website,
        'students': school.Number_of_Students,
        'type': school.type_of_school.Name if school.type_of_school else '',
        'settlement': {
            'name': school.settlement.Name if school.settlement else '',
            'type': school.settlement.Type if school.settlement else ''
        },
        'infrastructure': [i.Name for i in school.infrastructure],
        'specializations': [s.Name for s in school.specializations],
        'rating': school.get_avg_rating(),
        'review_count': school.get_review_count()
    }

    return jsonify(data)

# Отчеты (запросы из ТЗ)
```

```
# 1. Школы в Бийске типа "гимназия"
@app.route('/report/gymnasiums_bijsk')
@login_required
def report_gymnasiums_bijsk():
    bijsk = Settlement.query.filter(
        Settlement.Name.ilike('%бийск%'),
        Settlement.Type.ilike('%город%')
    ).first()

    if not bijsk:
        flash('Населенный пункт Бийск не найден', 'warning')
        return redirect(url_for('index'))

    gymnasium_type = TypeOfSchool.query.filter(
        TypeOfSchool.Name.ilike('%гимназия%')
    ).first()

    if not gymnasium_type:
        flash('Тип школы "гимназия" не найден', 'warning')
        return redirect(url_for('index'))

    schools = School.query.filter_by(
        PK_Settlement=bijsk.PK_Settlement,
        PK_Type_of_School=gymnasium_type.PK_Type_of_School,
        is_active=True
```

```
    ).order_by(School.Official_Name).all()

    return render_template('reports/gymnasiums_bijsk.html',
        schools=schools,
        settlement=bijsk,
        school_type=gymnasium_type)

# 2. Школы с библиотекой
@app.route('/report/schools_with_library')
@login_required
def report_schools_with_library():
    library = Infrastructure.query.filter(
        Infrastructure.Name.ilike('%библиотека%')
    ).first()

    if not library:
        flash('Инфраструктура "Библиотека" не найдена', 'warning')
        return redirect(url_for('index'))

    schools = School.query.filter(
        School.infrastructure.any(PK_Infrastructure=library.PK_Infrastructure),
        School.is_active == True
    ).order_by(School.Official_Name).all()

    return render_template('reports/schools_with_library.html',
```

```
        schools=schools,
        infrastructure=library)

# 3. Школы с углубленным изучением физики
@app.route('/report/schools_physics')
@login_required
def report_schools_physics():
    physics_spec = Specialization.query.filter(
        Specialization.Name.ilike('%физик%')
    ).first()

    schools = []
    if physics_spec:
        schools = School.query.filter(

School.specializations.any(PK_Specialization=physics_spec.PK_Specialization)
,
            School.is_active == True
        ).order_by(School.Official_Name).all()

    if not schools:
        physics_subject = Subject.query.filter(
            Subject.Name.ilike('%физик%')
        ).first()
```

```
        if physics_subject:
            schools = School.query.join(
                school_employee
            ).join(
                Employee
            ).join(
                employee_subject_competence
            ).filter(
                employee_subject_competence.c.PK_Subject ==
physics_subject.PK_Subject,
                School.is_active == True
            ).distinct().order_by(School.Official_Name).all()

    return render_template('reports/schools_physics.html',
        schools=schools)

# 4. Школы-интернаты с >200 учащихся
@app.route('/report/boarding_schools')
@login_required
def report_boarding_schools():
    boarding_type = TypeOfSchool.query.filter(
        TypeOfSchool.Name.ilike('%интернат%')
    ).first()

    if not boarding_type:
```

```
        flash('Тип школы "интернат" не найден', 'warning')
        return redirect(url_for('index'))

    schools = School.query.filter(
        School.PK_Type_of_School == boarding_type.PK_Type_of_School,
        School.Number_of_Students > 200,
        School.is_active == True
    ).order_by(db.desc(School.Number_of_Students)).all()

    return render_template('reports/boarding_schools.html',
        schools=schools,
        school_type=boarding_type)

# 5. Кадровый состав школы
@app.route('/report/school_staff/<int:school_id>')
@login_required
def report_school_staff(school_id):
    school = School.query.get_or_404(school_id)
    employees = school.employees

    return render_template('reports/school_staff.html',
        school=school,
        employees=employees)

# 6. Программы школы
```

```
@app.route('/report/school_programs/<int:school_id>')
@login_required
def report_school_programs(school_id):
    school = School.query.get_or_404(school_id)
    programs = school.programs

    return render_template('reports/school_programs.html',
        school=school,
        programs=programs)

# Аутентификация
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()

        if user and user.check_password(form.password.data):
            if not user.is_active:
                flash('Аккаунт деактивирован', 'danger')
                return redirect(url_for('login'))
```

```
            login_user(user, remember=form.remember_me.data)
            user.last_login = datetime.now(timezone.utc)
            db.session.commit()

            flash(f'Добро пожаловать, {user.username}!', 'success')

            next_page = request.args.get('next')
            return redirect(next_page or url_for('index'))

        flash('Неверное имя пользователя или пароль', 'danger')

    return render_template('auth/login.html', form=form)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))

    form = RegistrationForm()

    if form.validate_on_submit():
        # Проверка минимальной длины пароля
        if len(form.password.data) < 6:
            flash('Пароль должен содержать не менее 6 символов', 'danger')
            return render_template('auth/register.html', form=form)
```

```
        user = User(
            username=form.username.data,
            email=form.email.data,
            role=int(form.role.data)
        )

        # Если роль - работник учреждения, устанавливаем school_id
        if form.role.data == '2' and form.school_id.data and form.school_id.data !=
0:
            user.school_id = form.school_id.data

        user.set_password(form.password.data)

        db.session.add(user)
        db.session.commit()

        flash('Регистрация успешна! Теперь вы можете войти в систему.',
'success')
        return redirect(url_for('login'))

    # Показать ошибки валидации
    for field, errors in form.errors.items():
        for error in errors:
            flash(f'{getattr(form, field).label.text}: {error}', 'danger')
```

```
    return render_template('auth/register.html', form=form)


@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('Вы вышли из системы', 'info')
    return redirect(url_for('index'))


# Заглушка для госуслуг
@app.route('/auth/gosuslugi')
def auth_gosuslugi():
    flash('Интеграция с госуслугами находится в разработке', 'info')
    return redirect(url_for('login'))


@app.route('/auth/gosuslugi/callback')
def gosuslugi_callback():
    flash('Callback от госуслуг получен', 'info')
    return redirect(url_for('index'))


# Профиль пользователя
@app.route('/profile')
@login_required
def profile():
```

```
    form = UserProfileForm(
        username=current_user.username,
        email=current_user.email
    )

    user_reviews = Review.query.filter_by(user_id=current_user.id).count()

    return render_template('auth/profile.html',
        form=form,
        user_reviews=user_reviews)

@app.route('/profile/update', methods=['POST'])
@login_required
def update_profile():
    form = UserProfileForm()

    if form.validate_on_submit():
        if form.new_password.data:
            if not current_user.check_password(form.current_password.data):
                flash('Текущий пароль неверен', 'danger')
                return redirect(url_for('profile'))

            current_user.set_password(form.new_password.data)
            flash('Пароль успешно изменен', 'success')
```

```
        if current_user.username != form.username.data:
            current_user.username = form.username.data

        if current_user.email != form.email.data:
            current_user.email = form.email.data

        db.session.commit()
        flash('Профиль успешно обновлен', 'success')
    else:
        for field, errors in form.errors.items():
            for error in errors:
                flash(f'{getattr(form, field).label.text}: {error}', 'danger')

    return redirect(url_for('profile'))

# Административная часть
@login_required
@role_required('school_admin')
def admin_dashboard():
    from models import School, Review

    # Статистика
    total_schools = School.query.count()
    total_students =
db.session.query(db.func.sum(School.Number_of_Students)).scalar() or 0
```

```
    pending_reviews = Review.query.filter_by(is_approved=False).count()

    stats = {
        'total_schools': total_schools,
        'total_students': total_students
    }

    # Последние действия с информацией о пользователях
    recent_audits =
db.session.query(AuditLog).order_by(db.desc(AuditLog.timestamp)).limit(20).a
ll()

    return render_template('admin/dashboard.html',
                           stats=stats,
                           recent_audits=recent_audits,
                           pending_reviews=pending_reviews)

# Управление школами
@app.route('/admin/schools')
@login_required
@role_required('school_admin')
def admin_schools():
    page = request.args.get('page', 1, type=int)
    per_page = request.args.get('per_page', 20, type=int)
    status = request.args.get('status', '')
```

```
search = request.args.get('search', '')

# Базовый запрос
query = School.query

# Фильтр по статусу
if status == 'active':
    query = query.filter_by(is_active=True)
elif status == 'inactive':
    query = query.filter_by(is_active=False)

# Поиск
if search:
    query = query.filter(
        db.or_(
            School.Official_Name.ilike(f'%{search}%'),
            School.Legal_Adress.ilike(f'%{search}%'),
            School.Phone.ilike(f'%{search}%')
        )
    )

# Сортировка по дате создания (новые сверху)
query = query.order_by(db.desc(School.created_at))

# Пагинация
```

```
    schools = query.paginate(page=page, per_page=per_page, error_out=False)

    # Статистика
    active_count = School.query.filter_by(is_active=True).count()
    inactive_count = School.query.filter_by(is_active=False).count()

    return render_template('admin/schools.html',
                           schools=schools,
                           active_count=active_count,
                           inactive_count=inactive_count)

@app.route('/admin/school/add', methods=['GET', 'POST'])
@login_required
def add_school():
    # Проверяем, что пользователь - администратор или супер-администратор
    from models import User
    user = User.query.get(current_user.id)

    if not user or user.role < 4:
        flash('Доступ запрещен. Только администраторы могут добавлять школы.', 'danger')
        return redirect(url_for('index'))
    form = SchoolForm()
```

```
    form.type_of_school.choices = [(t.PK_Type_of_School, t.Name)
                                   for t in
TypeOfSchool.query.order_by(TypeOfSchool.Name).all()]
    form.settlement.choices = [(s.PK_Settlement, f'{s.Type} {s.Name}')
                               for s in Settlement.query.order_by(Settlement.Name).all()]
    form.infrastructure.choices = [(i.PK_Infrastructure, i.Name)
                                   for i in
Infrastructure.query.order_by(Infrastructure.Name).all()]
    form.specializations.choices = [(s.PK_Specialization, s.Name)
                                    for s in
Specialization.query.order_by(Specialization.Name).all()]


    if form.validate_on_submit():
        school = School(
            Official_Name=form.official_name.data,
            Legal_Adress=form.legal_address.data,
            Phone=form.phone.data,
            Email=form.email.data,
            Website=form.website.data,
            Founding_Date=form.founding_date.data,
            Number_of_Students=form.number_of_students.data,
            License=form.license.data,
            Accreditation=form.accreditation.data,
            PK_Type_of_School=form.type_of_school.data,
            PK_Settlement=form.settlement.data,
            created_by=current_user.id
```

```
        )

        for infra_id in form.infrastructure.data:
            infra = Infrastructure.query.get(infra_id)
            if infra:
                school.infrastructure.append(infra)

        for spec_id in form.specializations.data:
            spec = Specialization.query.get(spec_id)
            if spec:
                school.specializations.append(spec)

        db.session.add(school)
        db.session.commit()
        save_school_version_on_create(school, current_user.id)

        audit_log(current_user.id, 'create', 'School', school.PK_School,
                  new_values=school.Official_Name)

        flash('Школа успешно добавлена', 'success')
        return redirect(url_for('index'))

    return render_template('admin/add_school.html', form=form)


@app.route('/admin/school/<int:school_id>/edit', methods=['GET', 'POST'])
```

```
@login_required
def edit_school(school_id):
    from models import School

    school = School.query.get_or_404(school_id)

    # Проверка прав доступа
    if not can_edit_school(school_id):
        flash('У вас нет прав для редактирования этой школы', 'danger')
        return redirect(url_for('school_detail', school_id=school_id))

    # Создаем форму и передаем объект школы для автозаполнения
    form = SchoolForm(obj=school)

    # Заполняем choices для выпадающих списков
    form.type_of_school.choices = [(t.PK_Type_of_School, t.Name)
                                   for t in
TypeOfSchool.query.order_by(TypeOfSchool.Name).all()]
    form.settlement.choices = [(s.PK_Settlement, f'{s.Type} {s.Name}')
                               for s in Settlement.query.order_by(Settlement.Name).all()]
    form.infrastructure.choices = [(i.PK_Infrastructure, i.Name)
                                   for i in
Infrastructure.query.order_by(Infrastructure.Name).all()]
    form.specializations.choices = [(s.PK_Specialization, s.Name)
                                    for s in
Specialization.query.order_by(Specialization.Name).all()]
```

```
    # ВРУЧНУЮ заполняем поля, которые не совпадают с именами в форме
    # Это нужно сделать перед form.validate_on_submit()
    if request.method == 'GET':
        form.official_name.data = school.Official_Name
        form.legal_address.data = school.Legal_Adress  # Legal_Adress -> legal_address
        form.phone.data = school.Phone
        form.email.data = school.Email
        form.website.data = school.Website
        form.founding_date.data = school.Founding_Date
        form.number_of_students.data = school.Number_of_Students
        form.license.data = school.License  # License -> license
        form.accreditation.data = school.Accreditation  # Accreditation -> accreditation
        form.type_of_school.data = school.PK_Type_of_School
        form.settlement.data = school.PK_Settlement
        form.infrastructure.data = [i.PK_Infrastructure for i in school.infrastructure]
        form.specializations.data = [s.PK_Specialization for s in school.specializations]

    if form.validate_on_submit():
        old_values = {
            'Official_Name': school.Official_Name,
        'Legal_Adress': school.Legal_Adress,
```

```
        'Phone': school.Phone,
        'Email': school.Email,
        'Website': school.Website,
        'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,
        'Number_of_Students': school.Number_of_Students,
        'License': school.License,
        'Accreditation': school.Accreditation,
        'PK_Type_of_School': school.PK_Type_of_School,
        'PK_Settlement': school.PK_Settlement,
        'is_active': school.is_active,
        # Также инфраструктуру и специализации
        'infrastructure': [infra.PK_Infrastructure for infra in school.infrastructure],
        'specializations': [spec.PK_Specialization for spec in
school.specializations]
        }


        # Обновляем данные школы из формы
        school.Official_Name = form.official_name.data
        school.Legal_Adress = form.legal_address.data
        school.Phone = form.phone.data
        school.Email = form.email.data
        school.Website = form.website.data
        school.Founding_Date = form.founding_date.data
        school.Number_of_Students = form.number_of_students.data
```

```
school.License = form.license.data
school.Accreditation = form.accreditation.data
school.PK_Type_of_School = form.type_of_school.data
school.PK_Settlement = form.settlement.data

# Обновляем инфраструктуру
school.infrastructure = []
for infra_id in form.infrastructure.data:
    infra = Infrastructure.query.get(infra_id)
    if infra:
        school.infrastructure.append(infra)

# Обновляем специализации
school.specializations = []
for spec_id in form.specializations.data:
    spec = Specialization.query.get(spec_id)
    if spec:
        school.specializations.append(spec)

db.session.commit()
save_school_version_on_update(school, old_values, current_user.id)

audit_log(current_user.id, 'update', 'School', school.PK_School,
          old_values=old_values,
          new_values={'Official_Name': school.Official_Name})
```

```
        flash('Школа успешно обновлена', 'success')
        return redirect(url_for('index'))

    return render_template('admin/edit_school.html', form=form, school=school)

@app.route('/admin/school/<int:school_id>/delete', methods=['POST'])
@login_required
@role_required('super_admin')
def delete_school(school_id):
    school = School.query.get_or_404(school_id)

    school.is_active = False
    db.session.commit()
    save_school_version_on_delete(school, current_user.id)

    audit_log(current_user.id, 'delete', 'School', school.PK_School,
              old_values={'Official_Name': school.Official_Name})

    flash('Школа деактивирована', 'success')
    return redirect(url_for('admin_schools'))

# Импорт данных
@app.route('/admin/import', methods=['GET', 'POST'])
@login_required
```

```
@role_required('region_admin')
def import_data():
    form = ImportForm()

    if form.validate_on_submit():
        file_type = form.file_type.data

        if file_type == 'csv':
            headers = [
                'Official_Name', 'Legal_Adress', 'Phone', 'Email',
                'Website', 'Founding_Date', 'Number_of_Students',
                'Type_of_School', 'Settlement'
            ]

            output = io.StringIO()
            writer = csv.writer(output)
            writer.writerow(headers)
            writer.writerow([
                'Пример: Средняя школа №1',
                'г. Барнаул, ул. Ленина, 1',
                '+7 (3852) 123456',
                'school1@example.com',
                'http://school1.edu.ru',
                '1950-09-01',
                '500',
```

```
            'общеобразовательная школа',
            'г. Барнаул'
        ])
        output.seek(0)

        response = make_response(output.getvalue())
        response.headers['Content-Disposition'] = 'attachment;
filename=school_import_template.csv'
        response.headers['Content-Type'] = 'text/csv'
        return response

    elif file_type == 'json':
        template = {
            "schools": [
                {
                    "Official_Name": "Средняя школа №1",
                    "Legal_Adress": "г. Барнаул, ул. Ленина, 1",
                    "Phone": "+7 (3852) 123456",
                    "Email": "school1@example.com",
                    "Website": "http://school1.edu.ru",
                    "Founding_Date": "1950-09-01",
                    "Number_of_Students": 500,
                    "Type_of_School": "общеобразовательная школа",
                    "Settlement": "г. Барнаул"
                }
```

```
            ]
        }

        response = make_response(json.dumps(template, ensure_ascii=False, indent=2))
        response.headers['Content-Disposition'] = 'attachment; filename=school_import_template.json'
        response.headers['Content-Type'] = 'application/json'
        return response

    return render_template('admin/import.html', form=form)


@app.route('/admin/import/upload', methods=['POST'])
@login_required
@role_required('region_admin')
def upload_import():
    if 'file' not in request.files:
        flash('Файл не выбран', 'danger')
        return redirect(url_for('import_data'))

    file = request.files['file']
    if file.filename == '':
        flash('Файл не выбран', 'danger')
        return redirect(url_for('import_data'))
```

```
    file_extension = file.filename.rsplit('.', 1)[1].lower() if '.' in file.filename else ''

    imported_count = 0
    errors = []

    try:
        if file_extension == 'csv':
            data, error = import_from_csv(file)
            if error:
                flash(f'Ошибка чтения CSV: {error}', 'danger')
                return redirect(url_for('import_data'))

        elif file_extension in ['xlsx', 'xls']:
            data, error = import_from_excel(file)
            if error:
                flash(f'Ошибка чтения Excel: {error}', 'danger')
                return redirect(url_for('import_data'))

        elif file_extension == 'json':
            try:
                data = json.loads(file.read().decode('utf-8'))
                if 'schools' not in data:
                    flash('Неверный формат JSON: отсутствует ключ "schools"',
'danger')
                    return redirect(url_for('import_data'))
```

```
                data = data['schools']
            except:
                flash('Ошибка чтения JSON файла', 'danger')
                return redirect(url_for('import_data'))
        else:
            flash('Разрешены только CSV, Excel и JSON файлы', 'danger')
            return redirect(url_for('import_data'))

        for i, row in enumerate(data, 1):
            try:
                validation_errors = validate_school_data(row)
                if validation_errors:
                    errors.append(f'Строка {i}: {", ".join(validation_errors)}')
                    continue

                # Поиск типа школы
                type_name = row.get('Type_of_School', '')
                school_type = TypeOfSchool.query.filter(
                    TypeOfSchool.Name.ilike(f'%{type_name}%')
                ).first()

                if not school_type:
                    errors.append(f'Строка {i}: Тип школы "{type_name}" не
найден')
                    continue
```

```
            # Поиск населенного пункта
            settlement_name = row.get('Settlement', '')
            settlement = Settlement.query.filter(
                Settlement.Name.ilike(f'%{settlement_name}%')
            ).first()

            if not settlement:
                errors.append(f'Строка {i}: Населенный пункт "{settlement_name}" не найден')
                continue

            school = School(
                Official_Name=row['Official_Name'],
                Legal_Adress=row['Legal_Adress'],
                Phone=row['Phone'],
                Email=row.get('Email'),
                Website=row.get('Website'),
                Number_of_Students=row.get('Number_of_Students'),
                PK_Type_of_School=school_type.PK_Type_of_School,
                PK_Settlement=settlement.PK_Settlement,
                created_by=current_user.id
            )

            if row.get('Founding_Date'):
```

```
            try:
                school.Founding_Date = datetime.strptime(
                    row['Founding_Date'], '%Y-%m-%d'
                ).date()
            except:
                pass

        db.session.add(school)
        imported_count += 1

    except Exception as e:
        errors.append(f'Строка {i}: {str(e)}')

db.session.commit()

import_history = ImportHistory(
    filename=file.filename,
    file_type=file_extension,
    imported_by=current_user.id,
    record_count=imported_count,
    errors='\n'.join(errors) if errors else None
)
db.session.add(import_history)
db.session.commit()
```

```
        if errors:
            flash(f'Импорт завершен с ошибками. Успешно: {imported_count}, ошибок: {len(errors)}', 'warning')
        else:
            flash(f'Успешно импортировано {imported_count} школ', 'success')

        return redirect(url_for('admin_schools'))

    except Exception as e:
        db.session.rollback()
        flash(f'Ошибка при импорте: {str(e)}', 'danger')
        return redirect(url_for('import_data'))

# Экспорт данных
@app.route('/export/schools')
@login_required
def export_schools():
    format_type = request.args.get('format', 'csv')
    schools = School.query.filter_by(is_active=True).all()

    data = []
    for school in schools:
        data.append({
            'ID': school.PK_School,
            'Название': school.Official_Name,
```

```
            'Адрес': school.Legal_Adress,
            'Телефон': school.Phone,
            'Email': school.Email or '',
            'Сайт': school.Website or '',
            'Учащихся': school.Number_of_Students or '',
            'Тип школы': school.type_of_school.Name if school.type_of_school
else '',
            'Населенный пункт': f'{school.settlement.Type}
{school.settlement.Name}' if school.settlement else '',
            'Район': school.settlement.district.Name if school.settlement and
school.settlement.district else ''
        })

    if format_type == 'excel':
        output, filename = export_to_excel(data)
        response = make_response(output.getvalue())
        response.headers['Content-Disposition'] = f'attachment;
filename={filename}'
        response.headers['Content-Type'] =
'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    else:
        output, filename = export_to_csv(data)
        response = make_response(output.getvalue().encode('utf-8'))
        response.headers['Content-Disposition'] = f'attachment;
filename={filename}'
        response.headers['Content-Type'] = 'text/csv; charset=utf-8'
```

```
    return response


# Модуль отзывов
@app.route('/school/<int:school_id>/review/add', methods=['POST'])
@login_required
def add_review(school_id):
    school = School.query.get_or_404(school_id)
    if not school.is_active:
        abort(404)


    form = ReviewForm()
    if form.validate_on_submit():
        # Исправленная проверка прав для автоматического одобрения
        is_approved = False
        if current_user.role >= 2:  # Учитель (2) и выше автоматически одобряются
            is_approved = True


        review = Review(
            Author=current_user.username,
            Text=form.text.data,
            Rating=form.rating.data,
            Date=datetime.now(timezone.utc).date(),
            PK_School=school_id,
```

```
            user_id=current_user.id,
            is_approved=is_approved
        )

        db.session.add(review)
        db.session.commit()

        if is_approved:
            flash('Отзыв успешно добавлен', 'success')
        else:
            flash('Отзыв добавлен и будет опубликован после модерации',
'success')
    else:
        # Добавим отладку, чтобы увидеть ошибки формы
        print("Ошибки формы:", form.errors)  # Для отладки
        flash('Ошибка при добавлении отзыва. Проверьте заполнение полей.',
'danger')

    return redirect(url_for('school_detail', school_id=school_id))


@app.route('/admin/review/<int:review_id>/moderate', methods=['POST'])
@login_required
@role_required('school_admin')
def moderate_review(review_id):
```

```
    review = Review.query.get_or_404(review_id)
    action = request.form.get('action')
    comment = request.form.get('comment', '')

    if action == 'approve':
        review.is_approved = True
        review.moderated_by = current_user.id
        review.moderated_at = datetime.now(timezone.utc)
        review.moderation_comment = comment
        flash('Отзыв одобрен', 'success')
    elif action == 'reject':
        review.is_approved = False
        review.moderated_by = current_user.id
        review.moderated_at = datetime.now(timezone.utc)
        review.moderation_comment = comment
        flash('Отзыв отклонен', 'info')

    db.session.commit()
    return redirect(url_for('admin_reviews', status='pending'))

# Резервное копирование
@app.route('/admin/backup')
@login_required
@role_required('super_admin')
def create_backup_route():
```

```
    try:
        # Получаем параметры подключения из конфигурации
        db_uri = app.config['SQLALCHEMY_DATABASE_URI']

        # Парсим URI PostgreSQL
        # Формат: postgresql://user:password@host:port/database
        if not db_uri.startswith('postgresql://'):
            flash('Поддерживается только PostgreSQL для создания бэкапов',
'warning')
            return redirect(url_for('admin_dashboard'))

        # Извлекаем параметры
        # Убираем префикс
        db_uri = db_uri.replace('postgresql://', '')

        # Разделяем пользователя/пароль и хост/порт/базу
        parts = db_uri.split('@')
        if len(parts) != 2:
            flash('Неверный формат строки подключения к БД', 'danger')
            return redirect(url_for('admin_dashboard'))

        user_pass, host_port_db = parts
        username, password = user_pass.split(':')

        # Разделяем хост:порт/база
```

```
host_port, database = host_port_db.split('/')
if ':' in host_port:
    host, port = host_port.split(':')
else:
    host, port = host_port, '5432'

# Создаем имя файла с временной меткой
timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
backup_filename = f'dump_{timestamp}.sql'
backup_path = os.path.join('backups', backup_filename)

# Создаем директорию backups если её нет
os.makedirs('backups', exist_ok=True)

# Устанавливаем переменную окружения PGPASSWORD
env = os.environ.copy()
env['PGPASSWORD'] = password

# Формируем команду pg_dump
cmd = [
    'pg_dump',
    '-h', host,
    '-p', port,
    '-U', username,
    '-d', database,
```

```
            '-f', backup_path,
            '--clean',  # Добавить команды для очистки базы
            '--if-exists',  # Использовать IF EXISTS при удалении объектов
            '--create',  # Добавить команду CREATE DATABASE
            '--no-owner',  # Не указывать владельца
            '--no-privileges'  # Не добавлять привилегии
        ]

        # Выполняем команду
        result = subprocess.run(cmd, env=env, capture_output=True, text=True)

        if result.returncode != 0:
            flash(f'Ошибка при создании бэкапа: {result.stderr}', 'danger')
            return redirect(url_for('admin_dashboard'))

        # Отправляем файл для скачивания
        from flask import send_file
        return send_file(
            backup_path,
            as_attachment=True,
            download_name=backup_filename,
            mimetype='application/sql'
        )

    except Exception as e:
```

```
        app.logger.error(f"Error creating backup: {str(e)}")
        flash(f'Ошибка при создании бэкапа: {str(e)}', 'danger')
        return redirect(url_for('admin_dashboard'))

# Новые отчеты по требованию ТЗ

@app.route('/reports')
@login_required
def reports_index():
    """Страница выбора типа отчета"""
    return render_template('reports/index.html')

@app.route('/report/students')
@login_required
@role_required('school_admin')
def report_students():
    """Статистика по количеству учащихся"""
    from models import School, District, TypeOfSchool, Settlement

    # Параметры фильтрации
    district_id = request.args.get('district_id')
    type_id = request.args.get('type_id')
    min_year = request.args.get('min_year', 2000, type=int)
    max_year = request.args.get('max_year', 2026, type=int)
```

```
    # Базовый запрос
    query = School.query.filter_by(is_active=True)

    if district_id:
        query = query.join(Settlement).filter(Settlement.PK_District == district_id)

    if type_id:
        query = query.filter(School.PK_Type_of_School == type_id)

    schools = query.all()

    # Общая статистика
    total_schools = len(schools)
    total_students = sum(s.Number_of_Students or 0 for s in schools)
    avg_students_per_school = total_students / total_schools if total_schools else
0

    # Находим школу с максимальным количеством учащихся
    max_school = None
    max_students = 0
    for school in schools:
        if school.Number_of_Students and school.Number_of_Students >
max_students:
            max_students = school.Number_of_Students
            max_school = school
```

```
    # Статистика по районам
    districts_stats = []
    all_districts = District.query.all()
    overall_total_students = db.session.query(db.func.sum(School.Number_of_Students)).filter(
        School.is_active == True
    ).scalar() or 0

    for district in all_districts:
        district_schools = [s for s in schools if s.settlement and s.settlement.PK_District == district.PK_District]

        if district_schools:
            district_total = sum(s.Number_of_Students or 0 for s in district_schools)
            district_avg = district_total / len(district_schools) if district_schools else 0
            percentage = (district_total / overall_total_students * 100) if overall_total_students else 0

            districts_stats.append({
                'district': district.Name,
                'schools_count': len(district_schools),
                'total_students': district_total,
                'avg_students': round(district_avg, 1),
                'percentage': round(percentage, 1)  # ДОБАВЛЯЕМ ЭТО!
```

```
        })

    # Статистика по типам школ
    types_stats = []
    all_types = TypeOfSchool.query.all()

    for school_type in all_types:
        type_schools = [s for s in schools if s.PK_Type_of_School ==
school_type.PK_Type_of_School]

        if type_schools:
            type_total = sum(s.Number_of_Students or 0 for s in type_schools)
            type_avg = type_total / len(type_schools) if type_schools else 0

            types_stats.append({
                'type': school_type.Name,
                'schools_count': len(type_schools),
                'total_students': type_total,
                'avg_students': round(type_avg, 1)
            })

    # Топ школ по количеству учащихся
    top_schools = sorted([s for s in schools if s.Number_of_Students],
                         key=lambda x: x.Number_of_Students or 0,
                         reverse=True)[:10]
```

```
    # Динамика по годам (если есть даты основания)
    yearly_stats = {}
    for school in schools:
        if school.Founding_Date:
            year = school.Founding_Date.year
            if year not in yearly_stats:
                yearly_stats[year] = {'count': 0, 'total_students': 0, 'avg_students': 0}

            yearly_stats[year]['count'] += 1
            yearly_stats[year]['total_students'] += (school.Number_of_Students or
0)

    for year in yearly_stats:
        data = yearly_stats[year]
        data['avg_students'] = data['total_students'] / data['count'] if data['count'] >
0 else 0

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()

    return render_template('reports/students.html',
                           districts_stats=districts_stats,
                           types_stats=types_stats,
```

```
                           yearly_stats=yearly_stats,
                           top_schools=top_schools,
                           total_schools=total_schools,
                           total_students=total_students,
                           avg_students_per_school=round(avg_students_per_school, 1),
                           max_school=max_school,
                           max_students=max_students,
                           districts=districts,
                           school_types=school_types)

# Отчет по инфраструктуре с фильтрами
@app.route('/report/infrastructure')
@login_required
@role_required('school_admin')
def report_infrastructure():
    """Отчет по оснащенности школ"""
    from models import Infrastructure, School, District, Settlement,
TypeOfSchool

    # Получаем параметры фильтрации
    district_id = request.args.get('district_id')
    school_type_id = request.args.get('school_type_id')
    settlement_id = request.args.get('settlement_id')

    # Базовый запрос
```

```
query = School.query.filter_by(is_active=True)

# Применяем фильтры
if district_id:
    query = query.join(Settlement).filter(Settlement.PK_District == district_id)

if school_type_id:
    query = query.filter(School.PK_Type_of_School == school_type_id)

if settlement_id:
    query = query.filter(School.PK_Settlement == settlement_id)

schools = query.all()
total_schools = len(schools)

# Статистика по инфраструктуре
infrastructure_stats = []
all_infra = Infrastructure.query.all()

for infra in all_infra:
    count = School.query.filter(
        School.infrastructure.any(PK_Infrastructure=infra.PK_Infrastructure),
        School.is_active == True
    ).count()
```

```
        if district_id:
            count = len([s for s in schools if infra in s.infrastructure])

        percentage = (count / total_schools * 100) if total_schools else 0

        # Статистика по районам для этой инфраструктуры
        districts_data = []
        for district in District.query.all():
            district_count = School.query.join(Settlement).filter(
                School.infrastructure.any(PK_Infrastructure=infra.PK_Infrastructure),
                School.is_active == True,
                Settlement.PK_District == district.PK_District
            ).count()

            total_in_district = School.query.join(Settlement).filter(
                School.is_active == True,
                Settlement.PK_District == district.PK_District
            ).count()

            dist_percentage = (district_count / total_in_district * 100) if
total_in_district else 0
            districts_data.append({
                'name': district.Name,
                'percentage': round(dist_percentage, 1)
            })
```

```
        infrastructure_stats.append({
            'name': infra.Name,
            'count': count,
            'percentage': round(percentage, 1),
            'districts': districts_data
        })

    # Количество школ с конкретной инфраструктурой
    library = Infrastructure.query.filter_by(Name='Библиотека').first()
    gym = Infrastructure.query.filter_by(Name='Спортзал').first()
    lab = Infrastructure.query.filter_by(Name='Лаборатория').first()

    with_library = len([s for s in schools if library in s.infrastructure]) if library
else 0
    with_gym = len([s for s in schools if gym in s.infrastructure]) if gym else 0
    with_labs = len([s for s in schools if lab in s.infrastructure]) if lab else 0

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()
    settlements = Settlement.query.order_by(Settlement.Name).all()

    return render_template('reports/infrastructure.html',
                           schools=schools,
```

```
                           total_schools=total_schools,
                           with_library=with_library,
                           with_gym=with_gym,
                           with_labs=with_labs,
                           infrastructure_stats=infrastructure_stats,
                           districts=districts,
                           school_types=school_types,
                           settlements=settlements)

# Отчет по кадровому составу с фильтрами
@app.route('/report/staff')
@login_required
@role_required('school_admin')
def report_staff():
    """Отчет по кадровому составу"""
    from models import Employee, School, District, TypeOfSchool

    # Параметры фильтрации
    district_id = request.args.get('district_id')
    school_type_id = request.args.get('school_type_id')
    position_filter = request.args.get('position', '').lower()
    min_experience = request.args.get('min_experience', 0, type=int)

    # Базовый запрос
    query = Employee.query
```

```
# Фильтр по должности
if position_filter:
    if position_filter == 'директор':
        query = query.filter(Employee.Position.ilike('%директор%'))
    elif position_filter == 'завуч':
        query = query.filter(db.or_(
            Employee.Position.ilike('%завуч%'),
            Employee.Position.ilike('%заместитель%')
        ))
    elif position_filter == 'учитель':
        query = query.filter(Employee.Position.ilike('%учитель%'))

# Фильтр по стажу
if min_experience > 0:
    query = query.filter(Employee.Experience_Years >= min_experience)

employees = query.all()

# Если заданы фильтры по школе
if district_id or school_type_id:
    filtered_employees = []
    for emp in employees:
        # Получаем школы сотрудника
        emp_schools = School.query.filter(
```

```
            School.employees.any(PK_Employee=emp.PK_Employee),
            School.is_active == True
        )

        if district_id:
            emp_schools = emp_schools.join(Settlement).filter(Settlement.PK_District == district_id)

        if school_type_id:
            emp_schools = emp_schools.filter(School.PK_Type_of_School == school_type_id)

        if emp_schools.count() > 0:
            filtered_employees.append(emp)

    employees = filtered_employees

total_employees = len(employees)

# Статистика по должностям
position_stats = []
positions = {}

for emp in employees:
    pos = emp.Position.lower()
```

```
        key = None

        if 'директор' in pos:
            key = 'Директора'
        elif 'завуч' in pos or 'заместитель' in pos:
            key = 'Завучи'
        elif 'учитель' in pos or 'преподаватель' in pos:
            key = 'Учителя'
        else:
            key = 'Прочие'

        positions[key] = positions.get(key, 0) + 1

    for position, count in positions.items():
        percentage = (count / total_employees * 100) if total_employees else 0
        position_stats.append({
            'position': position,
            'count': count,
            'percentage': round(percentage, 1)
        })

    # Статистика по стажу
    experiences = [e.Experience_Years for e in employees if e.Experience_Years]
    avg_experience = round(sum(experiences) / len(experiences), 1) if
experiences else 0
```

```
    max_experience = max(experiences) if experiences else 0
    min_experience = min(experiences) if experiences else 0

    # Распределение по стажу
    experience_ranges = [
        {'range': '0-5 лет', 'min': 0, 'max': 5},
        {'range': '6-10 лет', 'min': 6, 'max': 10},
        {'range': '11-20 лет', 'min': 11, 'max': 20},
        {'range': '21-30 лет', 'min': 21, 'max': 30},
        {'range': 'Более 30 лет', 'min': 31, 'max': 100}
    ]

    for r in experience_ranges:
        count = len([e for e in employees if e.Experience_Years and r['min'] <=
e.Experience_Years <= r['max']])
        percentage = (count / total_employees * 100) if total_employees else 0
        r['count'] = count
        r['percentage'] = round(percentage, 1)

    # Количество по категориям
    directors_count = len([e for e in employees if 'директор' in
e.Position.lower()])
    deputies_count = len([e for e in employees if 'завуч' in e.Position.lower() or
'заместитель' in e.Position.lower()])
    teachers_count = len([e for e in employees if 'учитель' in e.Position.lower()
or 'преподаватель' in e.Position.lower()])
```

```
    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()

    return render_template('reports/staff.html',
                           employees=employees,
                           total_employees=total_employees,
                           directors_count=directors_count,
                           deputies_count=deputies_count,
                           teachers_count=teachers_count,
                           position_stats=position_stats,
                           avg_experience=avg_experience,
                           max_experience=max_experience,
                           min_experience=min_experience,
                           experience_ranges=experience_ranges,
                           districts=districts,
                           school_types=school_types)

# Отчет по программам дополнительного образования
@app.route('/report/programs')
@login_required
@role_required('school_admin')
def report_programs():
    """Отчет по образовательным программам"""
```

```
    from models import EducationProgram, School, District, TypeOfSchool,
Settlement


    # Параметры фильтрации

    district_id = request.args.get('district_id')

    school_type_id = request.args.get('school_type_id')

    program_type = request.args.get('program_type')


    # Базовый запрос для программ

    query = EducationProgram.query


    # Фильтр по типу программы

    if program_type:

        query = query.filter_by(Type=program_type)


    programs = query.all()


    # Статистика по программам

    program_stats = []

    for program in programs:

        # Школы с этой программой

        schools_query = School.query.filter(


School.programs.any(PK_Education_Program=program.PK_Education_Progra
m),

            School.is_active == True
```

```
        )

        # Применяем дополнительные фильтры
        if district_id:
            schools_query = schools_query.join(Settlement).filter(Settlement.PK_District == district_id)

        if school_type_id:
            schools_query = schools_query.filter(School.PK_Type_of_School == school_type_id)

        schools_with_program = schools_query.all()
        count = len(schools_with_program)

        total_schools = School.query.filter_by(is_active=True).count()
        percentage = (count / total_schools * 100) if total_schools else 0

        program_stats.append({
            'program': program,
            'schools_count': count,
            'percentage': round(percentage, 1),
            'schools': schools_with_program[:5]  # Первые 5 школ для примера
        })

    # Распределение по типам программ
```

```
    type_stats = []
    from collections import defaultdict
    program_types_count = defaultdict(int)

    for program in programs:
        program_types_count[program.Type] += 1

    for program_type, count in program_types_count.items():
        type_stats.append({
            'type': program_type,
            'count': count
        })

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()

    return render_template('reports/programs.html',
                           programs=programs,
                           program_stats=program_stats,
                           type_stats=type_stats,
                           districts=districts,
                           school_types=school_types)

# Экспорт отчетов
```

```
@app.route('/export/report')
@login_required
@role_required('school_admin')
def export_report():
    """Экспорт отчета в Excel/CSV"""
    report_type = request.args.get('report_type')
    format_type = request.args.get('format', 'excel')

    # В зависимости от типа отчета собираем данные
    # Здесь должна быть логика экспорта

    flash('Экспорт отчетов будет реализован в следующей версии', 'info')
    return redirect(url_for('reports_index'))
# Отчет по динамике набора
@app.route('/report/dynamics')
@login_required
@role_required('school_admin')
def report_dynamics():
    """Динамика набора учащихся по годам"""
    from models import School

    # Группировка по году основания
    yearly_stats = {}

    schools = School.query.filter(
```

```
        School.is_active == True,
        School.Founding_Date != None
    ).all()

    for school in schools:
        year = school.Founding_Date.year if school.Founding_Date else None
        if year:
            if year not in yearly_stats:
                yearly_stats[year] = {'count': 0, 'total_students': 0}

            yearly_stats[year]['count'] += 1
            yearly_stats[year]['total_students'] += (school.Number_of_Students or
0)

    # Рассчитываем среднее
    for year, data in yearly_stats.items():
        data['avg_students'] = data['total_students'] / data['count'] if data['count'] >
0 else 0

    return render_template('reports/dynamics.html',
                           yearly_stats=yearly_stats)

# Отчет по специализациям
@app.route('/report/specializations')
@login_required
```

```
@role_required('school_admin')
def report_specializations():
    """Отчет по специализациям школ"""
    from models import Specialization, School, District

    # Параметры фильтрации
    district_id = request.args.get('district_id')
    specialization_id = request.args.get('specialization_id')

    # Базовый запрос
    query = School.query.filter_by(is_active=True)

    if district_id:
        query = query.join(Settlement).filter(Settlement.PK_District == district_id)

    if specialization_id:
        query = query.filter(
            School.specializations.any(PK_Specialization=specialization_id)
        )

    schools = query.all()

    # Статистика по специализациям
    specialization_stats = []
    for spec in Specialization.query.all():
```

```
        count = School.query.filter(
            School.specializations.any(PK_Specialization=spec.PK_Specialization),
            School.is_active == True
        ).count()

        total_schools = School.query.filter_by(is_active=True).count()
        percentage = (count / total_schools * 100) if total_schools else 0

        specialization_stats.append({
            'specialization': spec,
            'count': count,
            'percentage': round(percentage, 1)
        })

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    specializations = Specialization.query.order_by(Specialization.Name).all()

    return render_template('reports/specializations.html',
                           schools=schools,
                           specialization_stats=specialization_stats,
                           districts=districts,
                           specializations=specializations)
@app.route('/api/settlements')
def get_all_settlements():
```

```
    settlements = Settlement.query.order_by(Settlement.Name).all()
    return jsonify([{
        'id': s.PK_Settlement,
        'name': s.Name,
        'type': s.Type
    } for s in settlements])


# История изменений школы
@app.route('/admin/school/<int:school_id>/history')
@login_required
@role_required('region_admin')
def school_history(school_id):
    school = School.query.get_or_404(school_id)

    # ВРЕМЕННЫЙ ФИКС - используем исправленный SQL прямо здесь
    sql = """
        SELECT
            sv.pk_version,
            sv.pk_school,
            sv.old_data,
            sv.new_data,
            sv.changed_at,
            sv.changed_by,
            sv.action,
            u.username
```

```
        FROM school_versions sv
        LEFT JOIN users u ON sv.changed_by = u.id
        WHERE sv.pk_school = :school_id
        ORDER BY sv.changed_at DESC
    """

    try:
        versions = db.session.execute(sql, {'school_id': school_id}).fetchall()
    except Exception as e:
        print(f"Ошибка получения версий: {e}")
        versions = []

    return render_template('admin/school_history.html',
                           school=school,
                           versions=versions)


# Управление проверками Рособрнадзора
@app.route('/admin/inspections')
@login_required
@role_required('region_admin')
def admin_inspections():
    """Управление проверками Рособрнадзора"""
    page = request.args.get('page', 1, type=int)
    status = request.args.get('status', 'all')
```

```
    query = Inspection.query

    if status == 'active':
        query = query.filter(Inspection.has_violations == True,
Inspection.is_resolved == False)
    elif status == 'resolved':
        query = query.filter(Inspection.is_resolved == True)
    elif status == 'with_violations':
        query = query.filter(Inspection.has_violations == True)

    inspections = query.order_by(Inspection.Date.desc()).paginate(page=page,
per_page=20)

    return render_template('admin/inspections.html',
                           inspections=inspections,
                           status=status)

@app.route('/admin/inspection/add', methods=['GET', 'POST'])
@login_required
@role_required('region_admin')
def add_inspection():
    """Добавление проверки"""
    from forms import InspectionForm
```

```
    form = InspectionForm()
    form.school_id.choices = [(s.PK_School, s.Official_Name)
                              for s in
School.query.filter_by(is_active=True).order_by(School.Official_Name).all()]

    if form.validate_on_submit():
        inspection = Inspection(
            Date=form.date.data,
            Result=form.result.data,
            Prescription_Number=form.prescription_number.data,
            PK_School=form.school_id.data,
            has_violations=form.has_violations.data,
            violation_type=form.violation_type.data,
            is_resolved=form.is_resolved.data,
            resolution_date=form.resolution_date.data,
            description=form.description.data
        )

        db.session.add(inspection)
        db.session.commit()

        audit_log(current_user.id, 'create', 'Inspection', inspection.PK_Inspection,
                  new_values={'Prescription_Number':
inspection.Prescription_Number})
```

```
        flash('Проверка успешно добавлена', 'success')
        return redirect(url_for('admin_inspections'))

    return render_template('admin/add_inspection.html', form=form)

@app.route('/admin/inspection/<int:inspection_id>/edit', methods=['GET',
'POST'])
@login_required
@role_required('region_admin')
def edit_inspection(inspection_id):
    """Редактирование проверки"""
    from forms import InspectionForm

    inspection = Inspection.query.get_or_404(inspection_id)
    form = InspectionForm(obj=inspection)

    form.school_id.choices = [(s.PK_School, s.Official_Name)
                              for s in
School.query.filter_by(is_active=True).order_by(School.Official_Name).all()]

    if request.method == 'GET':
        form.date.data = inspection.Date
        form.result.data = inspection.Result
        form.prescription_number.data = inspection.Prescription_Number
        form.school_id.data = inspection.PK_School
```

```
        form.has_violations.data = inspection.has_violations
        form.violation_type.data = inspection.violation_type
        form.is_resolved.data = inspection.is_resolved
        form.resolution_date.data = inspection.resolution_date
        form.description.data = inspection.description

    if form.validate_on_submit():
        old_values = {
            'Prescription_Number': inspection.Prescription_Number,
            'Result': inspection.Result[:50] + '...' if len(inspection.Result) > 50 else
inspection.Result
        }

        inspection.Date = form.date.data
        inspection.Result = form.result.data
        inspection.Prescription_Number = form.prescription_number.data
        inspection.PK_School = form.school_id.data
        inspection.has_violations = form.has_violations.data
        inspection.violation_type = form.violation_type.data
        inspection.is_resolved = form.is_resolved.data
        inspection.resolution_date = form.resolution_date.data
        inspection.description = form.description.data

        db.session.commit()
```

```
        audit_log(current_user.id, 'update', 'Inspection', inspection.PK_Inspection,
                  old_values=old_values,
                  new_values={'Prescription_Number':
inspection.Prescription_Number})

        flash('Проверка успешно обновлена', 'success')
        return redirect(url_for('admin_inspections'))

    return render_template('admin/edit_inspection.html', form=form,
inspection=inspection)

@app.route('/admin/inspection/<int:inspection_id>/delete', methods=['POST'])
@login_required
@role_required('super_admin')
def delete_inspection(inspection_id):
    """Удаление проверки"""
    inspection = Inspection.query.get_or_404(inspection_id)

    db.session.delete(inspection)
    db.session.commit()

    audit_log(current_user.id, 'delete', 'Inspection', inspection_id,
              old_values={'Prescription_Number': inspection.Prescription_Number})

    flash('Проверка удалена', 'success')
```

```
    return redirect(url_for('admin_inspections'))


# Удаление отзывов администратором
# В функции delete_review в app.py исправьте вызов audit_log:
@app.route('/admin/review/<int:review_id>/delete', methods=['POST'])
@login_required
@role_required('school_admin')
def delete_review(review_id):
    review = Review.query.get_or_404(review_id)
    school_id = review.PK_School

    # Исправленный вызов audit_log - убрать параметр description
    audit_log(current_user.id, 'delete', 'Review', review_id,
              old_values={
                  'Author': review.Author,
                  'Text': review.Text[:100] if review.Text else '',
                  'Rating': review.Rating,
                  'PK_School': review.PK_School
              })

    db.session.delete(review)
    db.session.commit()

    flash('Отзыв успешно удален', 'success')
    return redirect(url_for('school_detail', school_id=school_id))
```

```
@app.route('/admin/rollback/<int:version_id>', methods=['POST'])
@login_required
@role_required('super_admin')
def rollback_version(version_id):
    # Используем правильное имя поля
    version = SchoolVersion.query.get_or_404(version_id)

    # Получаем school_id из параметра запроса или из версии
    school_id = request.args.get('school_id')
    if not school_id:
        school_id = version.pk_school  # ИСПРАВЛЕНО: было version.school_id

    school = School.query.get_or_404(school_id)

    # Проверяем, что версия принадлежит этой школе
    if version.pk_school != school.PK_School:  # ИСПРАВЛЕНО: было version.school_id
        abort(403, "Эта версия не принадлежит указанной школе")

    # Получаем данные из старой версии
    if not version.old_data:
        flash('Нет данных для отката', 'danger')
        return redirect(url_for('school_history', school_id=school.PK_School))
```

```
    try:
        import json
        old_data = json.loads(version.old_data)

        # Сохраняем текущие значения перед откатом
        current_values = {
            'Official_Name': school.Official_Name,
            'Legal_Adress': school.Legal_Adress,
            'Phone': school.Phone,
            'Email': school.Email,
            'Website': school.Website,
            'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,
            'Number_of_Students': school.Number_of_Students,
            'License': school.License,
            'Accreditation': school.Accreditation,
            'PK_Type_of_School': school.PK_Type_of_School,
            'PK_Settlement': school.PK_Settlement,
            'is_active': school.is_active
        }

        # Восстанавливаем значения из старой версии
        school.Official_Name = old_data.get('Official_Name',
school.Official_Name)
```

```
        school.Legal_Adress = old_data.get('Legal_Adress', school.Legal_Adress)
        school.Phone = old_data.get('Phone', school.Phone)
        school.Email = old_data.get('Email', school.Email)
        school.Website = old_data.get('Website', school.Website)

        # Обрабатываем дату основания
        if old_data.get('Founding_Date'):
            from datetime import datetime
            try:
                school.Founding_Date =
datetime.fromisoformat(old_data['Founding_Date'])
            except ValueError:
                school.Founding_Date = old_data['Founding_Date']

        school.Number_of_Students = old_data.get('Number_of_Students',
school.Number_of_Students)
        school.License = old_data.get('License', school.License)
        school.Accreditation = old_data.get('Accreditation', school.Accreditation)
        school.PK_Type_of_School = old_data.get('PK_Type_of_School',
school.PK_Type_of_School)
        school.PK_Settlement = old_data.get('PK_Settlement',
school.PK_Settlement)
        school.is_active = old_data.get('is_active', school.is_active)

        # Восстанавливаем инфраструктуру
        if 'infrastructure' in old_data:
```

```
            school.infrastructure = []
            for infra_id in old_data['infrastructure']:
                infra = Infrastructure.query.get(infra_id)
                if infra:
                    school.infrastructure.append(infra)

        # Восстанавливаем специализации
        if 'specializations' in old_data:
            school.specializations = []
            for spec_id in old_data['specializations']:
                spec = Specialization.query.get(spec_id)
                if spec:
                    school.specializations.append(spec)

        db.session.commit()

        # Создаем запись об откате
        create_version('School', school.PK_School, 'rollback', current_values,
old_data, current_user.id)

        flash('Откат выполнен успешно', 'success')

    except Exception as e:
        db.session.rollback()
        flash(f'Ошибка при откате: {str(e)}', 'danger')
```

```
        print(f"Ошибка отката: {e}")

    return redirect(url_for('school_history', school_id=school.PK_School))

# Отчет по нарушениям Рособрнадзора
@app.route('/report/violations')
@login_required
def report_violations():
    """Отчет по нарушениям Рособрнадзора"""

    # Параметры фильтрации
    district_id = request.args.get('district_id', type=int)
    settlement_id = request.args.get('settlement_id', type=int)
    school_type_id = request.args.get('school_type_id', type=int)
    status = request.args.get('status', 'all')  # all, with_violations, resolved, active

    # Базовый запрос для инспекций
    query = Inspection.query.join(School)

    # Фильтры
    if district_id:
        query = query.join(Settlement).filter(Settlement.PK_District == district_id)

    if settlement_id:
        query = query.filter(School.PK_Settlement == settlement_id)
```

```
    if school_type_id:
        query = query.filter(School.PK_Type_of_School == school_type_id)

    # Фильтр по статусу нарушений
    if status == 'with_violations':
        query = query.filter(Inspection.has_violations == True)
    elif status == 'active':
        query = query.filter(Inspection.has_violations == True,
Inspection.is_resolved == False)
    elif status == 'resolved':
        query = query.filter(Inspection.has_violations == True,
Inspection.is_resolved == True)

    # Получаем данные
    inspections = query.order_by(Inspection.Date.desc()).all()

    # Данные для фильтров
    districts = District.query.order_by(District.Name).all()
    settlements = Settlement.query.order_by(Settlement.Name).all()
    school_types = TypeOfSchool.query.order_by(TypeOfSchool.Name).all()

    # Статистика
    total_inspections = len(inspections)
    with_violations = sum(1 for i in inspections if i.has_violations)
```

```
    resolved = sum(1 for i in inspections if i.has_violations and i.is_resolved)
    active = with_violations - resolved

    return render_template('reports/violations.html',
                           inspections=inspections,
                           districts=districts,
                           settlements=settlements,
                           school_types=school_types,
                           total_inspections=total_inspections,
                           with_violations=with_violations,
                           resolved=resolved,
                           active=active)


@app.cli.command('seed-history')
def seed_history_command():
    """Заполнение истории изменений тестовыми данными"""
    print("Заполнение истории изменений...")

    # Проверяем, есть ли школа с ID 102
    school = School.query.get(102)
    if not school:
        print("Школа с ID 102 не найдена")
        return
```

```
# Проверяем, есть ли пользователь администратор
admin = User.query.filter_by(username='admin').first()
if not admin:
    print("Пользователь 'admin' не найден")
    return

# Удаляем существующие версии для этой школы
from sqlalchemy import text
db.session.execute(
    text("DELETE FROM school_versions WHERE pk_school = :school_id"),
    {"school_id": 102}
)

# Создаем тестовые версии
from datetime import datetime
import json

# Версия 1: создание школы
db.session.execute(
    text("""
        INSERT INTO school_versions
        (pk_school, data_before, data_after, changed_at, changer_id, action)
        VALUES
        (:school_id, :data_before, :data_after, :changed_at, :changer_id, :action)
    """),
```

```
    {
        "school_id": 102,
        "data_before": None,
        "data_after": json.dumps({
            "Official_Name": "Барнаулская средняя школа",
            "Legal_Adress": "город Барнаул, ул. Песчаная, д. 769 стр. 84",
            "Phone": "+7 (3857) 728127",
            "Email": None,
            "Website": None,
            "Founding_Date": None,
            "Number_of_Students": None,
            "License": None,
            "Accreditation": None,
            "PK_Type_of_School": None,
            "PK_Settlement": None,
            "is_active": True
        }),
        "changed_at": datetime(2026, 1, 11, 15, 44, 0),
        "changer_id": admin.id,
        "action": "create"
    }
)


# Версия 2: обновление школы
db.session.execute(
```

```
text("""
    INSERT INTO school_versions
    (pk_school, data_before, data_after, changed_at, changer_id, action)
    VALUES
    (:school_id, :data_before, :data_after, :changed_at, :changer_id, :action)
"""),
{
    "school_id": 102,
    "data_before": json.dumps({
        "Official_Name": "Барнаулская средняя школа",
        "Legal_Adress": "город Барнаул, ул. Песчаная, д. 769 стр. 84",
        "Phone": "+7 (3857) 728127",
        "Email": None,
        "Website": None,
        "Founding_Date": None,
        "Number_of_Students": None,
        "License": None,
        "Accreditation": None,
        "PK_Type_of_School": None,
        "PK_Settlement": None,
        "is_active": True
    }),
    "data_after": json.dumps({
        "Official_Name": "Барнаулская средняя школаа",
        "Legal_Adress": "город Барнаул, ул. Песчаная, д. 769 стр. 84",
```

```
            "Phone": "+7 (3857) 728127",
            "Email": "school102@altai.edu.ru",
            "Website": "http://school102.altai.edu.ru",
            "Founding_Date": "1992-03-19",
            "Number_of_Students": "1233",
            "License": "Лицензия №Л035-35795 от 11.01.2026",
            "Accreditation": "Аккредитация №А417 от 11.01.2026",
            "PK_Type_of_School": "6",
            "PK_Settlement": "1",
            "is_active": "True"
        }),
        "changed_at": datetime(2026, 1, 11, 21, 56, 37),
        "changer_id": admin.id,
        "action": "update"
    }
)

    db.session.commit()
    print("✅ История изменений заполнена тестовыми данными")
# Модерация отзывов
@app.route('/admin/reviews')
@login_required
def admin_reviews():
    from models import User, Review
```

```
    user = User.query.get(current_user.id)

    # Проверяем права
    if not user or user.role < 2:
        flash('Доступ запрещен.', 'danger')
        return redirect(url_for('index'))

    try:
        page = request.args.get('page', 1, type=int)
        status = request.args.get('status', 'pending')

        # Если пользователь - работник учреждения, фильтруем по его школе
        if user.role == 2 and user.school_id:
            query = Review.query.filter_by(PK_School=user.school_id)
        else:
            query = Review.query

        if status == 'pending':
            query = query.filter_by(is_approved=False)
        elif status == 'approved':
            query = query.filter_by(is_approved=True)
        elif status == 'rejected':
            query =
query.filter_by(is_approved=False).filter(Review.rejection_reason.isnot(None))
```

```
        reviews = query.order_by(Review.Date.desc()).paginate(page=page,
per_page=20, error_out=False)


        if user.role == 2 and user.school_id:
            pending_count = Review.query.filter_by(is_approved=False,
PK_School=user.school_id).count()
            approved_count = Review.query.filter_by(is_approved=True,
PK_School=user.school_id).count()
        else:
            pending_count = Review.query.filter_by(is_approved=False).count()
            approved_count = Review.query.filter_by(is_approved=True).count()


        return render_template('admin/reviews.html',
                               reviews=reviews,
                               pending_count=pending_count,
                               approved_count=approved_count,
                               status=status)
    except Exception as e:
        app.logger.error(f"Error in admin_reviews: {str(e)}")
        flash(f'Ошибка при загрузке отзывов: {str(e)}', 'danger')
        return redirect(url_for('admin_dashboard'))


# Создание бэкапа


# Панель управления администратора
@app.route('/admin')
```

```
@login_required
def admin_dashboard():
    from models import User, School, Review, AuditLog
    from sqlalchemy import func

    user = User.query.get(current_user.id)

    # Проверяем, что пользователь имеет права администратора или работника учреждения
    if not user or user.role < 2:
        flash('Доступ запрещен. Только работники учреждений и администраторы имеют доступ к панели управления.', 'danger')
        return redirect(url_for('index'))

    try:
        # Статистика
        if user.role >= 4:  # Администраторы видят все школы
            total_schools = School.query.filter_by(is_active=True).count()
            total_students_result = db.session.query(func.sum(School.Number_of_Students)).filter_by(is_active=True).scalar()
        else:  # Работники учреждения видят только свою школу
            total_schools = 1
            total_students_result = School.query.filter_by(PK_School=user.school_id).first().Number_of_Students if user.school_id else 0
```

```
        total_students = total_students_result if total_students_result else 0


        # Отзывы на модерации
        if user.role >= 4:
            pending_reviews = Review.query.filter_by(is_approved=False).count()
        else:
            pending_reviews = Review.query.filter_by(is_approved=False,
PK_School=user.school_id).count()


        stats = {
            'total_schools': total_schools,
            'total_students': total_students
        }


        # Последние действия
        if user.role >= 4:
            recent_audits =
AuditLog.query.order_by(AuditLog.timestamp.desc()).limit(20).all()
        else:
            # Работники учреждения видят только действия, связанные с их
школой
            recent_audits = AuditLog.query.filter(
                (AuditLog.table_name == 'School') &
                (AuditLog.record_id == user.school_id)
            ).order_by(AuditLog.timestamp.desc()).limit(20).all()
```

```
        return render_template('admin/dashboard.html',
                               stats=stats,
                               recent_audits=recent_audits,
                               pending_reviews=pending_reviews)
    except Exception as e:
        app.logger.error(f"Error in admin_dashboard: {str(e)}")
        flash(f'Ошибка при загрузке панели управления: {str(e)}', 'danger')
        return redirect(url_for('index'))

# Функции для проверки прав доступа
def can_edit_school(school_id):
    """Проверяет, может ли текущий пользователь редактировать школу"""
    if not current_user.is_authenticated:
        return False

    # Получаем пользователя с текущим контекстом
    from models import User
    user = User.query.get(current_user.id)

    if not user:
        return False

    # Ученики/родители и другие не могут редактировать
    if user.role in [1, 3]:
```

```
        return False

    # Работники учреждения могут редактировать только свою школу
    if user.role == 2:
        return user.school_id == school_id

    # Админы могут редактировать все
    if user.role >= 4:
        return True

    return False


def school_employee_required(f):
    """Декоратор для проверки, что пользователь - работник учреждения"""
    @wraps(f)
    @login_required
    def decorated_function(*args, **kwargs):
        from models import User
        user = User.query.get(current_user.id)

        if not user or user.role != 2:
            flash('Доступ разрешен только работникам образовательных учреждений', 'danger')
            return redirect(url_for('index'))
```

```
        return f(*args, **kwargs)
    return decorated_function
# Команды CLI
@app.cli.command('init-db')
def init_db_command():
    """Инициализация базы данных"""
    print("Инициализация базы данных...")

    # Создаем таблицы
    db.create_all()

    # Создаем типы школ
    if not TypeOfSchool.query.first():
        types = [
            TypeOfSchool(Name='Общеобразовательная школа'),
            TypeOfSchool(Name='Гимназия'),
            TypeOfSchool(Name='Лицей'),
            TypeOfSchool(Name='Школа-интернат'),
            TypeOfSchool(Name='Коррекционная школа'),
            TypeOfSchool(Name='Вечерняя школа'),
            TypeOfSchool(Name='Кадетская школа')
        ]
        for t in types:
            db.session.add(t)
```

```
# Создаем районы
if not District.query.first():
    districts = [
        District(Name='Алейский район'),
        District(Name='Барнаульский район'),
        District(Name='Бийский район'),
        District(Name='Заринский район'),
        District(Name='Каменский район'),
        District(Name='Новоалтайский район'),
        District(Name='Рубцовский район'),
        District(Name='Славгородский район')
    ]
    for d in districts:
        db.session.add(d)

# Создаем населенные пункты
if not Settlement.query.first():
    settlements = [
        Settlement(Name='Барнаул', Type='город', PK_District=2),
        Settlement(Name='Бийск', Type='город', PK_District=3),
        Settlement(Name='Рубцовск', Type='город', PK_District=7),
        Settlement(Name='Новоалтайск', Type='город', PK_District=6),
        Settlement(Name='Заринск', Type='город', PK_District=4),
        Settlement(Name='Камень-на-Оби', Type='город', PK_District=5),
        Settlement(Name='Славгород', Type='город', PK_District=8),
```

```
        Settlement(Name='Алейск', Type='город', PK_District=1)
    ]
    for s in settlements:
        db.session.add(s)

# Создаем инфраструктуру
if not Infrastructure.query.first():
    infrastructures = [
        Infrastructure(Name='Спортзал'),
        Infrastructure(Name='Бассейн'),
        Infrastructure(Name='Библиотека'),
        Infrastructure(Name='Лаборатория'),
        Infrastructure(Name='Компьютерный класс'),
        Infrastructure(Name='Актовый зал'),
        Infrastructure(Name='Столовая'),
        Infrastructure(Name='Медицинский кабинет'),
        Infrastructure(Name='Спортивная площадка'),
        Infrastructure(Name='Мастерские')
    ]
    for i in infrastructures:
        db.session.add(i)

# Создаем специализации
if not Specialization.query.first():
    specializations = [
```

```
        Specialization(Name='Физико-математическая'),
        Specialization(Name='Химико-биологическая'),
        Specialization(Name='Гуманитарная'),
        Specialization(Name='Лингвистическая'),
        Specialization(Name='Техническая'),
        Specialization(Name='Художественно-эстетическая'),
        Specialization(Name='Спортивная'),
        Specialization(Name='Информационные технологии')
    ]
    for s in specializations:
        db.session.add(s)

# Создаем предметы
if not Subject.query.first():
    subjects = [
        Subject(Name='Математика'),
        Subject(Name='Физика'),
        Subject(Name='Химия'),
        Subject(Name='Биология'),
        Subject(Name='Русский язык'),
        Subject(Name='Литература'),
        Subject(Name='История'),
        Subject(Name='Обществознание'),
        Subject(Name='География'),
        Subject(Name='Иностранный язык'),
```

```
        Subject(Name='Информатика')
    ]
    for s in subjects:
        db.session.add(s)

# Создаем администратора
if not User.query.filter_by(username='admin').first():
    admin = User(
        username='admin',
        email='admin@example.com',
        role=5
    )
    admin.set_password('admin123')
    db.session.add(admin)

# Создаем тестовых пользователей
test_users = [
    ('parent', 'parent@example.com', 'parent123', 1),
    ('teacher', 'teacher@example.com', 'teacher123', 2),
    ('school_admin', 'school_admin@example.com', 'admin123', 3),
    ('region_admin', 'region_admin@example.com', 'admin123', 4)
]

for username, email, password, role in test_users:
    if not User.query.filter_by(username=username).first():
```

```
            user = User(username=username, email=email, role=role)
            user.set_password(password)
            db.session.add(user)

    db.session.commit()

    print('✅ База данных инициализирована с тестовыми данными.')
    print('👤 Администратор: admin / admin123')
    print('👨‍👩‍👧 Родитель: parent / parent123')
    print('👨‍🏫 Учитель: teacher / teacher123')
    print('🏫 Администратор школы: school_admin / admin123')
    print('🗺️ Администратор региона: region_admin / admin123')

@app.cli.command('create-user')
def create_user_command():
    """Создание нового пользователя"""
    import getpass

    username = input('Имя пользователя: ')
    if User.query.filter_by(username=username).first():
        print(f'❌ Пользователь {username} уже существует.')
        return

    email = input('Email: ')
    if User.query.filter_by(email=email).first():
```

```
    print(f'❌ Email {email} уже зарегистрирован.')
    return

print('Роли:')
print(' 0 - Гость')
print(' 1 - Родитель')
print(' 2 - Учитель')
print(' 3 - Администратор школы')
print(' 4 - Администратор региона')
print(' 5 - Супер-администратор')

try:
    role = int(input('Роль (0-5): '))
    if role < 0 or role > 5:
        print('❌ Роль должна быть от 0 до 5')
        return
except ValueError:
    print('❌ Роль должна быть числом')
    return

password = getpass.getpass('Пароль: ')
password2 = getpass.getpass('Повторите пароль: ')

if password != password2:
    print('❌ Пароли не совпадают')
```

```
        return

    if len(password) < 6:
        print('❌ Пароль должен быть не менее 6 символов')
        return

    user = User(username=username, email=email, role=role)
    user.set_password(password)

    db.session.add(user)
    db.session.commit()

    print(f'✅ Пользователь {username} создан с ролью {role}')

# Запуск приложения
if __name__ == '__main__':
    with app.app_context():
        try:
            db.create_all()
            print("✅ Database tables ready")
        except Exception as e:
            print(f"❌ Error: {e}")

        print("🚀 Starting School Registry application...")
        print("🌐 Server: http://localhost:5000")
```

print("📝 Press Ctrl+C to stop")

app.run(debug=True, host='0.0.0.0', port=5000)

config.py

```
import os
from datetime import timedelta
from dotenv import load_dotenv

load_dotenv()

basedir = os.path.abspath(os.path.dirname(__file__))

class Config:
    SECRET_KEY = os.environ.get('SECRET_KEY') or
'dev-secret-key-change-in-production'

    # PostgreSQL конфигурация
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
        'postgresql://postgres:password@localhost:5432/school_registry'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    SQLALCHEMY_ENGINE_OPTIONS = {
        'pool_pre_ping': True,
        'pool_recycle': 300,
    }

    # Настройки загрузки файлов
    UPLOAD_FOLDER = os.path.join(basedir, 'uploads')
    MAX_CONTENT_LENGTH = 16 * 1024 * 1024

    # Сессии
    PERMANENT_SESSION_LIFETIME = timedelta(hours=2)
```

```
    SESSION_COOKIE_HTTPONLY = True
    SESSION_COOKIE_SAMESITE = 'Lax'

    # Безопасность
    WTF_CSRF_ENABLED = True
    WTF_CSRF_SECRET_KEY = SECRET_KEY

    # Роли пользователей
    ROLES = {
        'guest': 0,
        'parent': 1,
        'teacher': 2,
        'school_admin': 3,
        'region_admin': 4,
        'super_admin': 5
    }

    # Настройки приложения
    SCHOOLS_PER_PAGE = 20
    DEFAULT_PAGE = 1

    # Интеграция с госуслугами (заглушки)
    GOSUSLUGI_APP_ID = os.environ.get('GOSUSLUGI_APP_ID',
'demo-app-id')
    GOSUSLUGI_REDIRECT_URI = os.environ.get('GOSUSLUGI_REDIRECT_URI',
        'http://localhost:5000/auth/gosuslugi/callback')

    @staticmethod
    def init_app(app):
        # Создаем необходимые директории
        directories = [
            app.config['UPLOAD_FOLDER'],
            os.path.join(app.config['UPLOAD_FOLDER'], 'backups'),
```

```
            os.path.join(app.config['UPLOAD_FOLDER'], 'imports'),
            os.path.join(basedir, 'instance'),
            os.path.join(basedir, 'static', 'css'),
            os.path.join(basedir, 'static', 'js'),
            os.path.join(basedir, 'static', 'images')
        ]
        for directory in directories:
            try:
                if not os.path.exists(directory):
                    os.makedirs(directory, exist_ok=True)
            except Exception:
                pass


class DevelopmentConfig(Config):
    DEBUG = True
    SQLALCHEMY_ECHO = False


    @staticmethod
    def init_app(app):
        Config.init_app(app)


class TestingConfig(Config):
    TESTING = True
    SQLALCHEMY_DATABASE_URI =
'postgresql://localhost/school_registry_test'
    WTF_CSRF_ENABLED = False


class ProductionConfig(Config):
    DEBUG = False
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL')
    if not SQLALCHEMY_DATABASE_URI:
        raise ValueError("DATABASE_URL environment variable is required
for production")
```

```
    SESSION_COOKIE_SECURE = True

    @classmethod
    def init_app(cls, app):
        Config.init_app(app)

        # Логирование в production
        import logging
        from logging.handlers import RotatingFileHandler

        if not app.debug:
            if not os.path.exists('logs'):
                os.mkdir('logs')
            file_handler = RotatingFileHandler(
                'logs/school_registry.log',
                maxBytes=10240,
                backupCount=10
            )
            file_handler.setFormatter(logging.Formatter(
                '%(asctime)s %(levelname)s: %(message)s [in
%(pathname)s:%(lineno)d]'
            ))
            file_handler.setLevel(logging.INFO)
            app.logger.addHandler(file_handler)
            app.logger.setLevel(logging.INFO)
            app.logger.info('School Registry startup')

config = {
    'development': DevelopmentConfig,
    'testing': TestingConfig,
    'production': ProductionConfig,
    'default': DevelopmentConfig
}
```

forms.py `from flask_wtf import FlaskForm`

```
from flask_wtf.file import FileField, FileAllowed

from wtforms import (

    StringField, PasswordField, BooleanField, SubmitField,

    TextAreaField, IntegerField, DateField, SelectField,

    SelectMultipleField, EmailField, URLField

)

from wtforms.validators import DataRequired, Length, EqualTo,
ValidationError, Optional, Email, URL

from models import User

from flask_wtf import FlaskForm

from wtforms import StringField, PasswordField, SubmitField,
SelectField, BooleanField, TextAreaField

from wtforms.validators import DataRequired, Email, EqualTo, Length,
ValidationError

from models import User, School



class LoginForm(FlaskForm):

    username = StringField('Имя пользователя',
validators=[DataRequired()])

    password = PasswordField('Пароль', validators=[DataRequired()])

    remember_me = BooleanField('Запомнить меня')

    submit = SubmitField('Войти')



class RegistrationForm(FlaskForm):

    username = StringField('Имя пользователя', validators=[

        DataRequired(),

        Length(min=3, max=80, message='Имя пользователя должно быть от
3 до 80 символов')

    ])



    email = StringField('Email', validators=[

        DataRequired(),

        Email(),
```

```
        Length(max=120)
    ])

    password = PasswordField('Пароль', validators=[
        DataRequired(),
        Length(min=6, message='Пароль должен быть не менее 6 символов')
    ])

    password2 = PasswordField('Повторите пароль', validators=[
        DataRequired(),
        EqualTo('password', message='Пароли должны совпадать')
    ])

    role = SelectField('Выберите вашу роль', choices=[
        ('1', 'Ученик/Родитель (только просмотр)'),
        ('2', 'Работник образовательного учреждения'),
        ('3', 'Другое (только просмотр)')
    ], validators=[DataRequired()])

    school_id = SelectField('Выберите ваше учреждение', choices=[],
coerce=int)

    submit = SubmitField('Зарегистрироваться')

    def __init__(self, *args, **kwargs):
        super(RegistrationForm, self).__init__(*args, **kwargs)
        # Заполняем список школ
        schools =
School.query.filter_by(is_active=True).order_by(School.Official_Name).a
ll()
        self.school_id.choices = [(0, '-- Выберите учреждение --')] + [
            (s.PK_School, s.Official_Name) for s in schools
        ]
```

```
    def validate_username(self, field):
        if User.query.filter_by(username=field.data).first():
            raise ValidationError('Это имя пользователя уже занято')

    def validate_email(self, field):
        if User.query.filter_by(email=field.data).first():
            raise ValidationError('Этот email уже зарегистрирован')

    def validate_school_id(self, field):
        if self.role.data == '2' and (field.data == 0 or field.data is
None):
            raise ValidationError('Работник учреждения должен выбрать
образовательное учреждение')

class SchoolForm(FlaskForm):
    official_name = StringField(
        'Официальное название*',
        validators=[DataRequired(), Length(max=255)]
    )
    legal_address = TextAreaField(
        'Юридический адрес*',
        validators=[DataRequired()]
    )
    phone = StringField(
        'Телефон*',
        validators=[DataRequired(), Length(max=20)]
    )
    email = EmailField(
        'Email',
        validators=[Optional(), Email(), Length(max=100)]
    )
    website = URLField(
```

```
    'Веб-сайт',
    validators=[Optional(), URL(), Length(max=200)]
)
founding_date = DateField(
    'Дата основания',
    format='%Y-%m-%d',
    validators=[Optional()]
)
number_of_students = IntegerField(
    'Количество учащихся',
    validators=[Optional()]
)
license = TextAreaField('Лицензия')
accreditation = TextAreaField('Аккредитация')
type_of_school = SelectField(
    'Тип школы*',
    coerce=int,
    validators=[DataRequired()]
)
settlement = SelectField(
    'Населенный пункт*',
    coerce=int,
    validators=[DataRequired()]
)
infrastructure = SelectMultipleField(
    'Инфраструктура',
    coerce=int
)
specializations = SelectMultipleField(
    'Специализации',
    coerce=int
)
```

```
    submit = SubmitField('Сохранить')


class ReviewForm(FlaskForm):
    text = TextAreaField(
        'Текст отзыва*',
        validators=[DataRequired(), Length(min=10, max=1000)]
    )
    rating = SelectField(
        'Оценка*',
        choices=[
            (5, '5 - Отлично'),
            (4, '4 - Хорошо'),
            (3, '3 - Удовлетворительно'),
            (2, '2 - Плохо'),
            (1, '1 - Очень плохо')
        ],
        coerce=int,
        validators=[DataRequired()]
    )
    submit = SubmitField('Оставить отзыв')


class ImportForm(FlaskForm):
    file_type = SelectField(
        'Тип файла',
        choices=[('csv', 'CSV'), ('json', 'JSON'), ('excel', 'Excel')],
        validators=[DataRequired()]
    )
    submit = SubmitField('Загрузить шаблон')


class EmployeeForm(FlaskForm):
    full_name = StringField(
        'ФИО*',
```

```
        validators=[DataRequired(), Length(max=100)]
    )
    position = StringField(
        'Должность*',
        validators=[DataRequired(), Length(max=50)]
    )
    qualifications = TextAreaField('Квалификация')
    experience_years = IntegerField(
        'Стаж работы (лет)',
        validators=[Optional()]
    )
    subjects = SelectMultipleField(
        'Предметы',
        coerce=int
    )
    submit = SubmitField('Сохранить')


class ProgramForm(FlaskForm):
    code_designation = StringField(
        'Кодовое обозначение*',
        validators=[DataRequired(), Length(max=50)]
    )
    name = StringField(
        'Название программы*',
        validators=[DataRequired(), Length(max=255)]
    )
    type = SelectField(
        'Тип программы*',
        choices=[
            ('основная', 'Основная'),
            ('дополнительная', 'Дополнительная')
        ],
```

```
        validators=[DataRequired()]
    )
    description = TextAreaField('Описание')
    submit = SubmitField('Сохранить')


class UserProfileForm(FlaskForm):
    username = StringField(
        'Имя пользователя',
        validators=[DataRequired(), Length(min=3, max=64)]
    )
    email = EmailField(
        'Email',
        validators=[DataRequired(), Email()]
    )
    current_password = PasswordField(
        'Текущий пароль',
        validators=[Optional()]
    )
    new_password = PasswordField(
        'Новый пароль',
        validators=[Optional(), Length(min=6)]
    )
    new_password2 = PasswordField(
        'Повторите новый пароль',
        validators=[EqualTo('new_password')]
    )
    submit = SubmitField('Обновить профиль')


class InspectionForm(FlaskForm):
    date = DateField('Дата проверки', validators=[DataRequired()])
    result = TextAreaField('Результат проверки',
validators=[DataRequired(), Length(max=2000)])
```

```
    prescription_number = StringField('Номер предписания',
validators=[DataRequired(), Length(max=50)])

    school_id = SelectField('Школа', coerce=int,
validators=[DataRequired()])

    has_violations = BooleanField('Нарушения выявлены')

    violation_type = StringField('Тип нарушений',
validators=[Optional(), Length(max=200)])

    is_resolved = BooleanField('Нарушения устранены')

    resolution_date = DateField('Дата устранения',
validators=[Optional()])

    description = TextAreaField('Описание', validators=[Optional(),
Length(max=1000)])

    submit = SubmitField('Сохранить')
```

models.py

```
from datetime import datetime, timezone

from flask_sqlalchemy import SQLAlchemy

from flask_login import UserMixin

from werkzeug.security import generate_password_hash,
check_password_hash

import json


db = SQLAlchemy()


# Таблицы многие-ко-многим

school_infrastructure = db.Table(

    'School_Infrastructure',

    db.Column('PK_Infrastructure', db.BigInteger,
db.ForeignKey('Infrastructure.PK_Infrastructure'), primary_key=True),

    db.Column('PK_School', db.BigInteger,
db.ForeignKey('School.PK_School'), primary_key=True)

)


school_specialization = db.Table(

    'School_Specialization',
```

```
    db.Column('PK_Specialization', db.BigInteger,
db.ForeignKey('Specialization.PK_Specialization'), primary_key=True),

    db.Column('PK_School', db.BigInteger,
db.ForeignKey('School.PK_School'), primary_key=True)

)


school_employee = db.Table(

    'School_Employee',

    db.Column('PK_School', db.BigInteger,
db.ForeignKey('School.PK_School'), primary_key=True),

    db.Column('PK_Employee', db.BigInteger,
db.ForeignKey('Employee.PK_Employee'), primary_key=True)

)


employee_subject_competence = db.Table(

    'Employee_Subject_Competence',

    db.Column('PK_Subject', db.BigInteger,
db.ForeignKey('Subject.PK_Subject'), primary_key=True),

    db.Column('PK_Employee', db.BigInteger,
db.ForeignKey('Employee.PK_Employee'), primary_key=True)

)


school_program_implementation = db.Table(

    'School_Program_Implementation',

    db.Column('PK_School', db.BigInteger,
db.ForeignKey('School.PK_School'), primary_key=True),

    db.Column('PK_Education_Program', db.BigInteger,
db.ForeignKey('Education_Program.PK_Education_Program'),
primary_key=True)

)


class User(db.Model):

    __tablename__ = 'user'


    id = db.Column(db.Integer, primary_key=True)

    username = db.Column(db.String(80), unique=True, nullable=False)
```

```
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(256), nullable=False)
    role = db.Column(db.Integer, default=1)  # 1: ученик/родитель, 2: работник учреждения, 3: другое, 4: админ
    school_id = db.Column(db.Integer, db.ForeignKey('School.PK_School'), nullable=True)  # Для работников учреждения
    is_active = db.Column(db.Boolean, default=True)
    created_at = db.Column(db.DateTime, default=datetime.now(timezone.utc))
    last_login = db.Column(db.DateTime, nullable=True)

    # Связи
    school = db.relationship('School', backref='school_users', foreign_keys=[school_id])

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

    @property
    def is_authenticated(self):
        return True

    @property
    def is_anonymous(self):
        return False

    def get_id(self):
        return str(self.id)

    def has_role(self, role_name):
```

```
        roles = {
            'parent_student': 1,
            'school_employee': 2,
            'other': 3,
            'admin': 4,
            'super_admin': 5
        }
        return self.role >= roles.get(role_name, 0)

    def can_edit_school(self, school_id=None):
        # Ученики/родители не могут редактировать
        if self.role == 1 or self.role == 3:
            return False
        # Работники учреждения могут редактировать только свою школу
        if self.role == 2:
            return school_id == self.school_id
        # Админы могут редактировать все
        if self.role >= 4:
            return True
        return False

class School(db.Model):
    __tablename__ = 'School'

    PK_School = db.Column(db.BigInteger, primary_key=True)
    Official_Name = db.Column(db.String(255), nullable=False)
    Legal_Adress = db.Column(db.Text, nullable=False)
    Phone = db.Column(db.String(20), nullable=False)
    Email = db.Column(db.String(100))
    Website = db.Column(db.String(200))
    Founding_Date = db.Column(db.Date)
    Number_of_Students = db.Column(db.BigInteger)
```

```
    License = db.Column(db.Text)

    Accreditation = db.Column(db.Text)

    PK_Type_of_School = db.Column(db.BigInteger,
db.ForeignKey('Type_of_School.PK_Type_of_School'))

    PK_Settlement = db.Column(db.BigInteger,
db.ForeignKey('Settlement.PK_Settlement'))



    # Дополнительные поля для системы

    is_active = db.Column(db.Boolean, default=True)

    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    updated_at = db.Column(db.DateTime, default=datetime.utcnow,
onupdate=datetime.utcnow)

    created_by = db.Column(db.Integer, db.ForeignKey('user.id'))



    # Связи

    type_of_school = db.relationship('TypeOfSchool', backref='schools')

    settlement = db.relationship('Settlement', backref='schools')

    infrastructure = db.relationship('Infrastructure',
secondary=school_infrastructure, backref='schools')

    specializations = db.relationship('Specialization',
secondary=school_specialization, backref='schools')

    employees = db.relationship('Employee', secondary=school_employee,
backref='schools')

    programs = db.relationship('EducationProgram',
secondary=school_program_implementation, backref='schools')

    reviews = db.relationship('Review', backref='school', lazy=True)

    inspections = db.relationship('Inspection', backref='school',
lazy=True)



    def get_avg_rating(self):

        """Средний рейтинг школы"""

        if not self.reviews:

            return 0



        try:
```

```
            approved_reviews = [r for r in self.reviews if getattr(r,
'is_approved', True)]
            if not approved_reviews:
                return 0
            return sum(r.Rating for r in approved_reviews) /
len(approved_reviews)
        except Exception:
            # Если есть ошибка с is_approved, считаем все отзывы
            return sum(r.Rating for r in self.reviews) /
len(self.reviews)

    def get_review_count(self):
        """Количество одобренных отзывов"""
        return len([r for r in self.reviews if r.is_approved and not
r.is_deleted])

    def __repr__(self):
        return f'<School {self.Official_Name}>'

class TypeOfSchool(db.Model):
    __tablename__ = 'Type_of_School'

    PK_Type_of_School = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(255), nullable=False)

    def __repr__(self):
        return f'<TypeOfSchool {self.Name}>'

class Settlement(db.Model):
    __tablename__ = 'Settlement'

    PK_Settlement = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(100), nullable=False)
    Type = db.Column(db.String(20), nullable=False)
```

```
    PK_District = db.Column(db.BigInteger,
db.ForeignKey('District.PK_District'))


    district = db.relationship('District', backref='settlements')


    def __repr__(self):
        return f'<Settlement {self.Type} {self.Name}>'


class District(db.Model):
    __tablename__ = 'District'


    PK_District = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(255), nullable=False)


    def __repr__(self):
        return f'<District {self.Name}>'


class Infrastructure(db.Model):
    __tablename__ = 'Infrastructure'


    PK_Infrastructure = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(50), nullable=False)


    def __repr__(self):
        return f'<Infrastructure {self.Name}>'


class Specialization(db.Model):
    __tablename__ = 'Specialization'


    PK_Specialization = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(50), nullable=False)


    def __repr__(self):
```

```
        return f'<Specialization {self.Name}>'


class Employee(db.Model):
    __tablename__ = 'Employee'

    PK_Employee = db.Column(db.BigInteger, primary_key=True)
    Full_Name = db.Column(db.String(100), nullable=False)
    Position = db.Column(db.String(50), nullable=False)

    # Дополнительные поля
    Qualifications = db.Column(db.Text, nullable=True)
    Experience_Years = db.Column(db.Integer, nullable=True)

    def __repr__(self):
        return f'<Employee {self.Full_Name}>'


class Subject(db.Model):
    __tablename__ = 'Subject'

    PK_Subject = db.Column(db.BigInteger, primary_key=True)
    Name = db.Column(db.String(100), nullable=False)

    def __repr__(self):
        return f'<Subject {self.Name}>'


class EducationProgram(db.Model):
    __tablename__ = 'Education_Program'

    PK_Education_Program = db.Column(db.BigInteger, primary_key=True)
    Code_Designation = db.Column(db.String(50), nullable=False)
    Name = db.Column(db.String(255), nullable=False)
    Type = db.Column(db.String(20), nullable=False)
```

```
    def __repr__(self):
        return f'<EducationProgram {self.Name}>'


class Review(db.Model):
    __tablename__ = 'Review'


    PK_Review = db.Column(db.BigInteger, primary_key=True)
    Author = db.Column(db.String(100))
    Text = db.Column(db.Text)
    Date = db.Column(db.Date, nullable=False)
    Rating = db.Column(db.Integer, nullable=False)
    PK_School = db.Column(db.BigInteger,
db.ForeignKey('School.PK_School'))


    # Поля системы
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    is_approved = db.Column(db.Boolean, default=True)
    moderated_by = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=True)
    moderated_at = db.Column(db.DateTime, nullable=True)
    moderation_comment = db.Column(db.Text, nullable=True)
    is_deleted = db.Column(db.Boolean, default=False)
    deleted_by = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=True)
    deleted_at = db.Column(db.DateTime, nullable=True)
    deletion_reason = db.Column(db.Text, nullable=True)


    def __repr__(self):
        return f'<Review {self.PK_Review} - School {self.PK_School}>'


class Inspection(db.Model):
    __tablename__ = 'Inspection'
```

```
    PK_Inspection = db.Column(db.BigInteger, primary_key=True)
    Date = db.Column(db.Date, nullable=False)
    Result = db.Column(db.Text, nullable=False)
    Prescription_Number = db.Column(db.String(50), nullable=False)
    PK_School = db.Column(db.BigInteger,
db.ForeignKey('School.PK_School'))

    # Дополнительные поля для отчетов
    has_violations = db.Column(db.Boolean, default=False)
    violation_type = db.Column(db.String(200), nullable=True)
    is_resolved = db.Column(db.Boolean, default=False)
    resolution_date = db.Column(db.Date, nullable=True)
    description = db.Column(db.Text, nullable=True)

    def __repr__(self):
        return f'<Inspection {self.Prescription_Number}>'

class AuditLog(db.Model):
    __tablename__ = 'audit_log'

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    action = db.Column(db.String(50), nullable=False)
    table_name = db.Column(db.String(50), nullable=False)
    record_id = db.Column(db.String(100), nullable=False)
    old_values = db.Column(db.Text, nullable=True)
    new_values = db.Column(db.Text, nullable=True)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)
    ip_address = db.Column(db.String(45))
    user_agent = db.Column(db.Text, nullable=True)

    def __repr__(self):
```

```
        return f'<AuditLog {self.action}
{self.table_name}.{self.record_id}>'


class ImportHistory(db.Model):

    __tablename__ = 'import_history'


    id = db.Column(db.Integer, primary_key=True)

    filename = db.Column(db.String(255), nullable=False)

    file_type = db.Column(db.String(10), nullable=False)

    imported_by = db.Column(db.Integer, db.ForeignKey('user.id'))

    imported_at = db.Column(db.DateTime, default=datetime.utcnow)

    record_count = db.Column(db.Integer)

    status = db.Column(db.String(20), default='completed')

    errors = db.Column(db.Text, nullable=True)


    def __repr__(self):

        return f'<ImportHistory {self.filename}>'


class DataVersion(db.Model):

    __tablename__ = 'data_versions'


    pk_version = db.Column(db.Integer, primary_key=True)

    table_name = db.Column(db.String(100), nullable=False)

    record_id = db.Column(db.Integer, nullable=False)

    action = db.Column(db.String(20), nullable=False)

    data_before = db.Column(db.JSON, nullable=True)

    data_after = db.Column(db.JSON, nullable=True)

    changed_by = db.Column(db.Integer, db.ForeignKey('users.id'))

    changed_at = db.Column(db.DateTime,
default=datetime.now(timezone.utc))

    created_at = db.Column(db.DateTime,
default=datetime.now(timezone.utc))
```

```
    def __repr__(self):
        return f'<DataVersion {self.pk_version}:
{self.table_name}.{self.record_id} - {self.action}>'

    def get_changes(self):
        """Возвращает список измененных полей"""
        if not self.data_before or not self.data_after:
            return []

        changes = []
        for key in set(list(self.data_before.keys()) +
list(self.data_after.keys())):
            before = self.data_before.get(key)
            after = self.data_after.get(key)
            if before != after:
                changes.append({
                    'field': key,
                    'before': before,
                    'after': after
                })
        return changes

# Найти определение модели SchoolVersion и исправить ForeignKey
class SchoolVersion(db.Model):
    __tablename__ = 'school_versions'

    pk_version = db.Column(db.Integer, primary_key=True)
    pk_school = db.Column(db.Integer,
db.ForeignKey('School.PK_School'), nullable=False)
    version_number = db.Column(db.Integer, nullable=False)
    action = db.Column(db.String(50), nullable=False)
    old_data = db.Column(db.Text)
    new_data = db.Column(db.Text)
```

```
    changed_by = db.Column(db.Integer, db.ForeignKey('user.id'))
    changed_at = db.Column(db.DateTime, default=datetime.utcnow)

    # ИСПРАВЬТЕ ЭТИ СТРОКИ:
    # school = db.relationship('School', backref=db.backref('versions', lazy='dynamic'))
    # user = db.relationship('User', backref=db.backref('versions', lazy='dynamic'))

    # СТАЛО:
    school = db.relationship('School', backref=db.backref('school_versions', lazy='dynamic'))
    user = db.relationship('User', backref=db.backref('school_versions_created', lazy='dynamic'))
```

requirements.txt

```
Flask==2.3.3
Flask-SQLAlchemy==3.0.5
Flask-Login==0.6.3
Flask-WTF==1.2.1
WTForms==3.0.1
python-dotenv==1.0.0
Flask-Migrate==4.0.5
SQLAlchemy==1.4.49
psycopg2-binary
Werkzeug==2.3.7
email-validator==2.1.0
pandas
openpyxl==3.0.10
faker
```

utils.py

```
import os
import json
import csv
```

```
import io

import pandas as pd

from datetime import datetime

from functools import wraps

from flask import current_app, request, abort

from models import db, AuditLog

from models import DataVersion

from datetime import datetime, timezone

import json

# utils.py

from models import db, School, Review


def create_version(table_name, record_id, action, data_before,
data_after, user_id):

    """Создание записи о версии данных"""

    from models import SchoolVersion, db  # Импортируем SchoolVersion,
а не DataVersion

    from datetime import datetime, timezone

    import json


    print(f"DEBUG: create_version вызвана для {table_name} {record_id},
действие: {action}")


    if data_before:

        data_before_json = json.dumps(data_before, ensure_ascii=False,
default=str)

    else:

        data_before_json = None


    if data_after:

        data_after_json = json.dumps(data_after, ensure_ascii=False,
default=str)

    else:

        data_after_json = None
```

```
    # Используем SchoolVersion вместо DataVersion
    version = SchoolVersion(
        pk_school=record_id,  # или другое поле в зависимости от таблицы
        version_number=1,  # нужно реализовать подсчет версий
        action=action,
        old_data=data_before_json,
        new_data=data_after_json,
        changed_by=user_id,
        changed_at=datetime.now(timezone.utc)
    )

    print(f"DEBUG: Создана версия: {version}")

    try:
        db.session.add(version)
        db.session.commit()
        print(f"DEBUG: Версия сохранена в БД")
    except Exception as e:
        print(f"ERROR: Ошибка сохранения версии: {e}")
        db.session.rollback()

    return version




def save_school_version_on_create(school, user_id):
    """Сохранение версии при создании школы"""
    school_data = {
        'Official_Name': school.Official_Name,
        'Legal_Adress': school.Legal_Adress,
```

```
        'Phone': school.Phone,
        'Email': school.Email,
        'Website': school.Website,
        'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,
        'Number_of_Students': school.Number_of_Students,
        'License': school.License,
        'Accreditation': school.Accreditation,
        'PK_Type_of_School': school.PK_Type_of_School,
        'PK_Settlement': school.PK_Settlement,
        'is_active': school.is_active
    }

    create_version('School', school.PK_School, 'create', None,
school_data, user_id)



def save_school_version_on_update(school, old_values, user_id):
    """Сохранение версии при обновлении школы"""
    # Получаем полные данные школы ДО изменения
    school_data_before = {
        'Official_Name': old_values.get('Official_Name', '') if
old_values else school.Official_Name,
        'Legal_Adress': old_values.get('Legal_Adress', '') if
old_values else school.Legal_Adress,
        'Phone': old_values.get('Phone', '') if old_values else
school.Phone,
        'Email': old_values.get('Email', '') if old_values else
school.Email,
        'Website': old_values.get('Website', '') if old_values else
school.Website,
        'Founding_Date': old_values.get('Founding_Date', '') if
old_values else (school.Founding_Date.isoformat() if
school.Founding_Date else None),
        'Number_of_Students': old_values.get('Number_of_Students', '')
if old_values else school.Number_of_Students,
```

```
        'License': old_values.get('License', '') if old_values else
school.License,

        'Accreditation': old_values.get('Accreditation', '') if
old_values else school.Accreditation,

        'PK_Type_of_School': old_values.get('PK_Type_of_School', '') if
old_values else school.PK_Type_of_School,

        'PK_Settlement': old_values.get('PK_Settlement', '') if
old_values else school.PK_Settlement,

        'is_active': old_values.get('is_active', True) if old_values
else school.is_active

    }


    # Полные данные школы ПОСЛЕ изменения

    school_data_after = {

        'Official_Name': school.Official_Name,

        'Legal_Adress': school.Legal_Adress,

        'Phone': school.Phone,

        'Email': school.Email,

        'Website': school.Website,

        'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,

        'Number_of_Students': school.Number_of_Students,

        'License': school.License,

        'Accreditation': school.Accreditation,

        'PK_Type_of_School': school.PK_Type_of_School,

        'PK_Settlement': school.PK_Settlement,

        'is_active': school.is_active

    }


    create_version('School', school.PK_School, 'update',
school_data_before, school_data_after, user_id)


def save_school_version_on_delete(school, user_id):

    """Сохранение версии при удалении школы"""

    school_data = {
```

```
        'Official_Name': school.Official_Name,
        'Legal_Adress': school.Legal_Adress,
        'Phone': school.Phone,
        'Email': school.Email,
        'Website': school.Website,
        'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,
        'Number_of_Students': school.Number_of_Students,
        'License': school.License,
        'Accreditation': school.Accreditation,
        'PK_Type_of_School': school.PK_Type_of_School,
        'PK_Settlement': school.PK_Settlement,
        'is_active': school.is_active
    }

    create_version('School', school.PK_School, 'delete', school_data,
None, user_id)




def get_versions(table_name, record_id, limit=50):
    """Получить историю изменений записи"""
    return DataVersion.query.filter_by(
        table_name=table_name,
        record_id=record_id
    ).order_by(db.desc(DataVersion.changed_at)).limit(limit).all()


def rollback_to_version(version_id):
    """Откат к конкретной версии"""
    version = DataVersion.query.get(version_id)
    if not version:
        return False, "Версия не найдена"
```

```
    if version.action == 'delete' and version.data_before:
        # Восстановление удаленной записи
        # Здесь нужно импортировать соответствующую модель
        # и создать объект из data_before
        pass
    elif version.action in ['create', 'update'] and
version.data_before:
        # Восстановление данных из предыдущей версии
        # Здесь нужно обновить запись данными из data_before
        pass

    # Создаем запись об откате
    create_version(
        table_name=version.table_name,
        record_id=version.record_id,
        action='rollback',
        data_before=version.data_after,
        data_after=version.data_before,
        user_id=current_user.id
    )

    return True, "Откат выполнен успешно"


def get_school_versions(school_id, limit=50):
    """Получить историю изменений школы"""
    return get_versions('School', school_id, limit)


def save_school_version(school, action, user_id):
    """Сохранение версии школы"""
    school_data = {
        'Official_Name': school.Official_Name,
        'Legal_Adress': school.Legal_Adress,
        'Phone': school.Phone,
```

```
        'Email': school.Email,
        'Website': school.Website,
        'Founding_Date': school.Founding_Date.isoformat() if
school.Founding_Date else None,
        'Number_of_Students': school.Number_of_Students,
        'License': school.License,
        'Accreditation': school.Accreditation,
        'PK_Type_of_School': school.PK_Type_of_School,
        'PK_Settlement': school.PK_Settlement,
        'is_active': school.is_active,
        'infrastructure': [infra.PK_Infrastructure for infra in
school.infrastructure],
        'specializations': [spec.PK_Specialization for spec in
school.specializations]
    }

    # Для update нужно получить предыдущую версию
    if action == 'update':
        last_version = DataVersion.query.filter_by(
            table_name='School',
            record_id=school.PK_School,
            action='update'
        ).order_by(db.desc(DataVersion.changed_at)).first()

        if last_version:
            data_before = last_version.data_after
        else:
            # Если нет предыдущей версии update, берем create версию
            create_version = DataVersion.query.filter_by(
                table_name='School',
                record_id=school.PK_School,
                action='create'
            ).first()
```

```
            data_before = create_version.data_after if create_version
else school_data

    else:

        data_before = None


    create_version(

        table_name='School',

        record_id=school.PK_School,

        action=action,

        data_before=data_before,

        data_after=school_data,

        user_id=user_id

    )


def allowed_file(filename, allowed_extensions):

    """Проверка разрешенных расширений файлов"""

    return '.' in filename and \

           filename.rsplit('.', 1)[1].lower() in allowed_extensions


def audit_log(user_id, action, table_name, record_id, old_values=None,
new_values=None):

    """Запись в журнал аудита"""

    log = AuditLog(

        user_id=user_id,  # Должен быть ID пользователя, а не None

        action=action,

        table_name=table_name,

        record_id=record_id,

        old_values=json.dumps(old_values, ensure_ascii=False) if
old_values else None,

        new_values=json.dumps(new_values, ensure_ascii=False) if
new_values else None,

        timestamp=datetime.now(timezone.utc)  # Добавляем явно
timestamp

    )
```

```
    db.session.add(log)
    db.session.commit()


def role_required(role_name):
    """Декоратор для проверки ролей"""
    def decorator(f):
        @wraps(f)
        def decorated_function(*args, **kwargs):
            from flask_login import current_user
            if not current_user.is_authenticated:
                abort(401)
            if not current_user.has_role(role_name):
                abort(403)
            return f(*args, **kwargs)
        return decorated_function
    return decorator


def export_to_csv(data, filename=None):
    """Экспорт данных в CSV"""
    if not data:
        return None


    headers = list(data[0].keys())
    output = io.StringIO()
    writer = csv.DictWriter(output, fieldnames=headers)
    writer.writeheader()
    writer.writerows(data)
    output.seek(0)


    if not filename:
        filename =
f'export_{datetime.now().strftime("%Y%m%d_%H%M%S")}.csv'
```

```
    return output, filename


def export_to_excel(data, filename=None):
    """Экспорт данных в Excel"""
    if not data:
        return None

    df = pd.DataFrame(data)
    output = io.BytesIO()
    with pd.ExcelWriter(output, engine='openpyxl') as writer:
        df.to_excel(writer, index=False, sheet_name='Данные')
    output.seek(0)

    if not filename:
        filename =
f'export_{datetime.now().strftime("%Y%m%d_%H%M%S")}.xlsx'

    return output, filename


def import_from_csv(file_stream):
    """Импорт данных из CSV"""
    try:
        if hasattr(file_stream, 'read'):
            content = file_stream.read().decode('utf-8-sig')
        else:
            content = file_stream

        csv_file = io.StringIO(content)
        sample = csv_file.read(1024)
        csv_file.seek(0)
        delimiter = ',' if ',' in sample else ';'

        reader = csv.DictReader(csv_file, delimiter=delimiter)
```

```
        data = [row for row in reader]
        return data, None
    except Exception as e:
        return None, str(e)

def import_from_excel(file_stream):
    """Импорт данных из Excel"""
    try:
        df = pd.read_excel(file_stream)
        data = df.to_dict('records')
        return data, None
    except Exception as e:
        return None, str(e)

def validate_school_data(data):
    """Валидация данных школы"""
    errors = []
    required_fields = ['Official_Name', 'Legal_Adress', 'Phone']

    for field in required_fields:
        if not data.get(field):
            errors.append(f'Поле "{field}" обязательно для заполнения')

    if data.get('Email') and '@' not in data['Email']:
        errors.append('Неверный формат email')

    if data.get('Phone') and len(data['Phone']) > 20:
        errors.append('Телефон слишком длинный')

    if data.get('Number_of_Students'):
        try:
            students = int(data['Number_of_Students'])
```

```
            if students < 0:
                errors.append('Количество учащихся не может быть отрицательным')
        except ValueError:
            errors.append('Количество учащихся должно быть числом')

    return errors

def generate_report_stats():
    """Генерация статистики для админ-панели"""
    try:
        # Импортируем здесь, чтобы избежать циклических импортов
        from models import School, Review

        # Безопасно считаем школы и учащихся
        total_schools = School.query.filter_by(is_active=True).count()

        total_students_result = db.session.query(db.func.sum(School.Number_of_Students)).scalar()
        total_students = total_students_result or 0

        # Простой подсчет отзывов - не используем колонки, которых может не быть
        try:
            total_reviews = Review.query.count()
        except Exception:
            total_reviews = 0

        try:
            # Используем SQL, чтобы избежать проблем с несуществующими колонками
            from sqlalchemy import text
            result = db.session.execute(text("SELECT COUNT(*) FROM \"Review\""))
```

```
            total_reviews = result.scalar() or 0
        except Exception:
            total_reviews = 0

        # Для отзывов на модерации будем осторожнее
        pending_reviews = 0

        stats = {
            'total_schools': total_schools,
            'total_students': total_students,
            'total_reviews': total_reviews,
            'pending_reviews': pending_reviews
        }
        return stats
    except Exception as e:
        print(f"Ошибка при генерации статистики: {e}")
        return {
            'total_schools': 0,
            'total_students': 0,
            'total_reviews': 0,
            'pending_reviews': 0
        }

def create_backup():
    """Создание резервной копии базы данных"""
    try:
        backup_dir = os.path.join(current_app.config['UPLOAD_FOLDER'],
'backups')
        if not os.path.exists(backup_dir):
            os.makedirs(backup_dir)

        timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
```

```
        backup_file = os.path.join(backup_dir,
f'db_backup_{timestamp}.sql')

        # Для PostgreSQL используем pg_dump
        import subprocess
        db_url = current_app.config['SQLALCHEMY_DATABASE_URI']

        # Извлекаем данные из URL
        from urllib.parse import urlparse
        parsed = urlparse(db_url)
        dbname = parsed.path[1:]
        user = parsed.username
        password = parsed.password
        host = parsed.hostname
        port = parsed.port or 5432

        # Команда pg_dump
        env = os.environ.copy()
        env['PGPASSWORD'] = password

        cmd = [
            'pg_dump',
            '-h', host,
            '-p', str(port),
            '-U', user,
            '-d', dbname,
            '-f', backup_file
        ]

        result = subprocess.run(cmd, env=env, capture_output=True,
text=True)

        if result.returncode == 0:
```

```
            return backup_file
        else:
            current_app.logger.error(f'Ошибка при создании резервной
копии: {result.stderr}')
            return None

    except Exception as e:
        current_app.logger.error(f'Ошибка при создании резервной копии:
{e}')
        return None

def render_star_rating(rating):
    """Генерация HTML для звездного рейтинга"""
    full_stars = int(rating)
    half_star = rating - full_stars >= 0.5
    empty_stars = 5 - full_stars - (1 if half_star else 0)

    stars_html = ''
    stars_html += '<i class="bi bi-star-fill text-warning"></i>' *
full_stars
    if half_star:
        stars_html += '<i class="bi bi-star-half text-warning"></i>'
    stars_html += '<i class="bi bi-star text-warning"></i>' *
empty_stars

    return stars_html

def pluralize(number, singular, plural1, plural2=None):
    """Функция для склонения существительных"""
    if plural2 is None:
        plural2 = plural1

    if number % 10 == 1 and number % 100 != 11:
        return singular
```

```
    elif 2 <= number % 10 <= 4 and (number % 100 < 10 or number % 100
>= 20):
        return plural1
    else:
        return plural2
```