Московский Государственный Университет Геодезии и Картографии
(МИИГАиК)

Факультет геоинформатики и информационной безопасности

Кафедра информационно-измерительных систем

Разработка детализированной документации системного аналитика

«Система управления проектами для образовательных учреждений»

Работу выполнил:

Студент 2023-ФГиИБ-ИСиТ-2б

Смыслов Е. А.

Работу проверила:

Лазуренко Н. С.

Москва 2025

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Назначение проекта

Целью разработки является создание веб-ориентированной информационной системы для управления пользователями онлайн-магазина. Система демонстрирует применение архитектурного паттерна MVC (Model-View-Controller) и обеспечивает автоматизацию процессов администрирования клиентской базы.

1.2. Описание предметной области

Система решает задачи управления учетными записями в контексте электронной коммерции. Основные бизнес-процессы:

- Просмотр списка зарегистрированных пользователей.
- Регистрация новых клиентов в системе.
- Редактирование контактных данных (имя, email) и прав доступа.
- Удаление неактуальных учетных записей.

1.3. Основные требования к проекту

Функциональные требования:

- Реализация полного цикла CRUD (Create, Read, Update, Delete) для сущности «Пользователь».
- Валидация вводимых данных (обязательные поля, формат email).

Нефункциональные требования:

- **Архитектура:** Строгое следование паттерну MVC.
- **Интерфейс:** Адаптивный дизайн (Responsive Web Design) с использованием библиотеки Bootstrap 5.

- **Масштабируемость:** Модульная структура кода, позволяющая легко добавлять новые сущности (например, товары, заказы).

1.4. Основные требования к проекту

Предположения и ограничения

- **Язык программирования:** Python 3.10+.
- **Фреймворк:** Flask (микрофреймворк).
- **СУБД:** SQLite (в режиме разработки), с использованием ORM SQLAlchemy.

2. АРХИТЕКТУРНЫЕ РЕШЕНИЯ

2.1. Общая архитектура системы

Приложение построено как классическое серверное веб-приложение. Сервер обрабатывает HTTP-запросы от клиента (браузера), обращается к базе данных и возвращает готовые HTML-страницы.

2.2. Архитектура MVC

В проекте реализовано четкое разделение ответственности компонентов:

- **Model (Модель):** Файл models/user.py. Описывает структуру данных и правила взаимодействия с БД. Не содержит логики отображения.
- **View (Представление):** Папка templates/. HTML-шаблоны с использованием синтаксиса Jinja2 и стилей Bootstrap. Отвечают только за визуализацию данных, полученных от контроллера.
- **Controller (Контроллер):** Файл controllers/user_controller.py. Обрабатывает входящие запросы, вызывает методы модели и выбирает шаблон для ответа.

2.3. Обоснование выбора технологий

Python + Flask: Обеспечивают высокую скорость разработки и наглядность реализации паттерна MVC. Flask позволяет явно выделять маршруты (routes) и связывать их с функциями-контроллерами.

SQLAlchemy (ORM): Используется для абстрагирования от SQL-запросов. Это повышает безопасность (защита от SQL-инъекций) и позволяет легко сменить СУБД в будущем (например, на PostgreSQL).

Bootstrap 5: CSS-фреймворк, обеспечивающий современный внешний вид и кроссбраузерность без глубокого написания кастомных стилей.

3. МОДЕЛИРУЕМЫЕ СУЩНОСТИ (MODEL)

3.1. Структуры данных

Поле	Тип данных	Описание	Ограничения
id	Integer	Уникальный идентификатор	Primary Key, Auto Increment
name	String(100)	Имя пользователя	Not Null
email	String(100)	Электронная почта	Unique, Not Null
role	String(50)	Роль в системе	Default: 'customer'

3.2. Логика обработки данных

При создании объекта модели используется **Паттерн Factory** (через конструктор класса). ORM автоматически отслеживает изменения объектов (Unit of Work) и синхронизирует их с базой данных при вызове commit().

3.3. Хранение и обработка данных

Данные хранятся в реляционной базе данных SQLite (файл database.db). Взаимодействие осуществляется через сессию базы данных, что обеспечивает атомарность транзакций (если ошибка произойдет в середине операции, изменения не сохранятся).

4. ПРЕДСТАВЛЕНИЕ (VIEW)

4.1. Пользовательские интерфейсы

Интерфейс реализован с использованием шаблонного движка **Jinja2**.

- **Base Layout (base.html):** Содержит общую структуру (HTML head, навигационная панель Bootstrap, подключение скриптов). Использует механизм блоков ({% block content %}) для вставки содержимого конкретных страниц.
- **Список пользователей (index.html):** Отображает таблицу с данными. Реализованы кнопки действий с иконками и стилями Bootstrap (btn-primary, btn-danger).
- **Форма (user_form.html):** Единый шаблон для создания и редактирования. Использует компоненты Card и Form Control для эстетичного отображения.

4.2. Форматы вывода данных

Сервер генерирует HTML-код динамически. Данные передаются из контроллера в шаблон в виде объектов Python, где они перебираются в циклах ({% for user in users %}) и подставляются в разметку.

4.3. UI-функционал

1. Использованы модальные окна и алерты (JavaScript confirm) для подтверждения удаления записей.
2. Адаптивная верстка

5. КОНТРОЛЛЕРЫ (CONTROLLER)

5.1. Управление потоком действий

Контроллеры организованы с помощью Flask Blueprints, что обеспечивает модульность системы. **Алгоритм обработки запроса:**

1. API-шлюз (Flask Router) определяет URL запроса.
2. Вызывается соответствующая функция контроллера.
3. Контроллер анализирует метод (GET или POST).
4. В случае POST: данные извлекаются из request.form, валидируются и передаются Модели.
5. В случае GET: запрашиваются данные у Модели.
6. Возвращается результат функции render_template или redirect.

5.2. Реализация обработчиков событий

Основные маршруты системы:

- GET /users/ — получение списка всех объектов (Read).
- GET /users/create — отображение формы создания.
- POST /users/create — прием данных и создание записи (Create).
- POST /users/update/<id> — обновление существующей записи (Update).
- GET /users/delete/<id> — удаление записи (Delete).

5.3. Безопасность контроллеров

Защита от SQL-инъекций реализована средствами ORM.

Обработка исключений (try/except) предотвращает падение сервера при ошибках базы данных.

ЗАКЛЮЧЕНИЕ

В ходе практической работы была спроектирована и реализована информационная система управления онлайн-магазином. **Достигнутые результаты:**

1. Изучена и применена на практике архитектура **MVC**, что позволило разделить бизнес-логику и визуальную часть.
2. Освоена работа с микро-фреймворком Flask и ORM SQLAlchemy.
3. Реализованы структурные шаблоны проектирования при организации кода.
4. Создан современный пользовательский интерфейс с использованием Bootstrap 5.