Частное образовательное учреждение высшего образования

«Казанский инновационный университет имени В. Г. Тимирясова (ИЭУП)»

Факультет менеджмента и инженерного бизнеса

Кафедра информационных технологий и техносферной безопасности

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине «Интернет-программирование»**

**на тему: «Разработка веб-ориентированной информационной системы “Автокурсы”»**

Направление подготовки: 09.03.03 «Прикладная информатика»

**Выполнил:** \_\_\_\_\_\_\_\_\_\_\_

обучающийся гр. 1071(К) очной формы обучения Балабанов С.А.

**Руководитель**: \_\_\_\_\_\_\_\_\_

к.т.н., доцент каф. ИТиТБ Иванов В.А.

**Результат защиты:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Казань, 2025 г.**

Private Higher Education Institution

«Kazan Innovative University named after V. G. Timiryasov (IEML)» Faculty of Management and Engineering Business

Department of Information Technology and Technosphere Security

**COURSE PROJECT**

**on discipline «Web-programming»**

**on the topic: «Development of a web-based information system “Avtokursy”»**

Course of study: 09.03.03 «Applied Informatics»

**Student: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

learning gr. 1071(K) full-time study Balabanov S.A.

**Supervisor:** \_\_\_\_\_\_\_\_\_\_\_

Ph.D., assoc. prof. Dep. of ITaTS Ivanov V.A.

**Result of protection:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Kazan, 2025**

**АННОТАЦИЯ**

на курсовой проект

по дисциплине «Интернет-программирование»

|  |
| --- |
| Ф.И.О. студента: Балабанов Сергей Андреевич |
| Student’s Name: Balabanov Sergey Andreevich |
| Направление: 09.03.03 «Прикладная информатика» |
| № группы: 1071(К) |

|  |
| --- |
| Название курсового проекта: |
| Разработка веб-ориентированной информационной системы “Автокурсы” |
| Paper Title: |
| Development of a web-based information system “Avtokursy” |

Аннотация:

|  |
| --- |
| Данная курсовая работа посвязена разработке веб-ориентированной информационной системы «Автокурсы». Был проведён анализ требований пользователей, определена архитектура системы и реализована клиент-серверная модель с использованием фреймворка Django и СУБД PostgreSQL. Разработаны модули для регистрации пользователей, отображения курсов и управления ими. Создан адаптивный интерфейс с применением HTML, CSS и Bootstrap. Также обеспечено развертывание и настройка проекта на сервере. Разработанная система направлена на улучшение качества подготовки водителей и автоматизацию процессов взаимодействия между участниками обучения. |

Summary:

|  |
| --- |
| This course work is devoted to the development of a web-based information system "Avtokursy". An analysis of user requirements was carried out, the system architecture was determined, and a client-server model was implemented using the Django framework and the PostgreSQL database management system. Modules have been developed for user registration, course display, and course management. An adaptive interface has been created using HTML, CSS, and Bootstrap. The project is also deployed and configured on the server. The developed system is aimed at improving the quality of driver training and automating the processes of interaction between training participants. |

|  |  |
| --- | --- |
| Руководитель: к.т.н., доцент каф.  ИТиТБ Фасхутдинов Л.М. | Перевод проверил: |
| « » 2020 г. | « » 2020 г. |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc197268283)

[1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 7](#_Toc197268284)

[1.1 Описание предметной области 7](#_Toc197268285)

[1.2 Составление технического задания 8](#_Toc197268286)

[1.3 Проектирование архитектуры приложения (MTV) 11](#_Toc197268287)

[2. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ НА DJANGO 13](#_Toc197268288)

[2.1 Структура проекта и настройка среды разработки 13](#_Toc197268289)

[2.2 Реализация серверной части 15](#_Toc197268290)

[2.3 Реализация клиентской части 20](#_Toc197268291)

[3. РАЗВЁРТЫВАНИЕ И ЭКСПЛУАТАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 26](#_Toc197268292)

[ЗАКЛЮЧЕНИЕ 29](#_Toc197268293)

[СПИСОК ЛИТЕРАТУРЫ 31](#_Toc197268294)

[Приложение А. Листинг исходных кодов 32](#_Toc197268295)

# **ВВЕДЕНИЕ**

На сегодняшний день, в условиях повсеместной автоматизации и развития цифровых технологий, самым актуальным способом передачи информации и взаимодействия людей являются веб-ориентированные информационные системы. Цифровизация затронула многие сферы нашей жизни – банковский сектор, общение, документооборот и так далее. Однако одной из востребованных сфер остаётся обучение вождению транспортных средств.

В условиях роста населения в городах, а также доступность автотранспортных средств всё чаще люди обращаются в автошколы для обучения вождению транспортных средств. Многие автошколы до сих пор применяют устаревшие методы работы с клиентами и документооборотом. Значительная часть организации процесса обучения осуществляется вручную – с использованием бумажных журналов, телефонных записей, социальных сетей и таблиц. Данные методы приводят к ряду проблем, а именно к неэффективному распределению времени и ресурсов, отсутствию единого хранилища данных об учащихся и инструкторах, сложности с регистрацией новых учеников и слабому уровню обучения. Таким образом, можно выделить необходимость и актуальность создания современной веб-системы, направленной на сосредоточение множества различных курсов на одном информационном портале, автоматизацию записи на курсы, хранения информации о занятиях и пользователей, а также облегчённому взаимодействию инструкторов и курсантов.

Целью данной курсовой работы является разработка веб-ориентированной информационной системы «Автокурсы», предназначенной для сосредоточения курсов в одном веб-приложении, автоматизации процесса управления и взаимодействия между пользователями.

Чтобы достичь поставленную цель необходимо решить следующие задачи:

1. Проанализировать проблемные точки и определить функциональные технические требования к информационной системе.
2. Разработать архитектуру проекта, включая описание структуры базы данных и логики взаимодействия компонентов
3. Реализовать клиент-серверное веб-приложение с использованием фреймворка Django.
4. Создать пользовательский интерфейс.
5. Обеспечить развертывание и настройку системы на сервере.

В процессе реализации проекта были использованы следующие технические и программные средства:

1. Для реализации серверной стороны был выбран фреймворк Django для Python, обладающий встроенной поддержкой ORM, системы аутентификации и архитектуры MTV (Model-Template-View)
2. Для разработки клиентской части использованы HTML5, CSS3, а также фреймворк Bootstrap и язык программирования JavaScript, что позволяет обеспечить адаптивность и интерактивность пользовательского интерфейса.
3. Дизайн-макеты интерфейсов создавались в среде Figma
4. Для хранения данных применялась реляционная база данных PostgreSQL
5. Среда разработки: PyCharm, Git для контроля версий, Python v3.13

Таким образом, проведённый анализ показал, что сфера обучения вождению на данный момент требует цифровизации. В ходе работы над курсовой работой мы разработали веб-ориентированную систему «Автокурсы», на платформе Django как надёжный и проверенный фреймворк, позволяющий эффективно реализовать архитектуру MTV, которая ориентирована на улучшение управления процессами обучения, сосредоточит курсы в едином реестре, повысит удобство для пользователей, а также снизит высокую нагрузку на автошколы.

# **1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

## **1.1 Описание предметной области**

Современные автошколы часто ориентированы на минимальное соответствие требованиям для получения водительского удостоверения, однако они не уделяют внимания подготовки водителей к реальному движению. В результате многие выпускники таких школ чувствуют себя неуверенно на дорогах [3]. Поэтому растёт спрос на сервисы, где можно найти курсы по вождению в реальных условиях.

Поэтому веб-приложение «Автокурсы» предназначено для упрощения взаимодействия между курсантами и частными инструкторами.

Основными участниками системы являются:

1. Курсанты – пользователи, которые ищут дополнительные занятия по обучению вождения
2. Инструкторы – специалисты, которые создают курсы и предлагают свои услуги по обучению вождению

Ключевыми процессами являются:

1. Регистрация и авторизация пользователей с разными ролями – инструктор или ученик
2. Публикация курсов от инструкторов с описанием цен и опыта.
3. Возможность онлайн-записи на курс и отслеживания занятий

Данная система реализуется с использованием фреймворка Django для скорости и обеспечении безопасности взаимодействия, реляционной СУБД PostgreSQL, реализующее централизованное хранение данных о курсах, инструкторах и учениках, языков Python, HTML/CSS и JavaScript для построения клиент-серверного приложения.

Таким образом, описание предметной области позволило выделить ключевые сущности, взаимодействие курсантов и инструкторов, основные взаимосвязи и процессы. Эти данные служат основой для проектирования функциональной архитектуры системы «Автокурсы».

## **1.2 Составление технического задания**

Техническое задание (ТЗ) – это документ, в котором фиксируются требования к проекту. Техническое задание включает в себя точно прописанные характеристики итогового продукта. Оно служит основой для проектирования, разработки и внедрения продукта [4].

Что конкретно стоит добавлять в техническое задание зависит от конкретного продукта, однако есть некоторый свод общих правил для составления правильного задания:

* Общие положения. Включает в себя название проекта, цель разработки, область применения.
* Требование к графическому интерфейсу. Здесь указывается требования к дизайну, используемые UI/UX-инструменты.
* Эксплуатационные требования. Описывают условия, в которых будет функционировать система, а именно требования к серверу и хостингу, совместимость с браузерами и ОС.
* Функциональные требования. В этом разделе фиксируются основные модули системы, роли пользователей и их действия, поведение системы при разных сценариях.
* Требования к надежности и безопасности. Содержит меры защиты данных, устойчивость к сбоям и требования к нагрузке.
* Структура проекта. Это графическое или текстовое описание структуры сайта, а именно страницы и модули, маршруты и связи между модулями.

Также можно указать используемые технологии и программные средства, будущие планы по расширению, ограничения и риски. Таким образом, грамотно составленное ТЗ способствует успешной реализации всех этапов проекта. Теперь, благодаря исследованию предметной области и данного шаблона составления технического задания мы можем расписать собственное представление об веб-ориентированной информационной системы «Автокурсы».

1. Общие положения

Целью проекта является реализация информационной системы «Автокурсы». Данная система представляет собой веб-платформу, позволяющей курсантам находить и бронировать услуги частных инструкторов, а инструкторам предлагать свои курсы ученикам. Система должна предоставить удобный и понятный интерфейс, автоматизировать процесс подбора инструкторов, а также ведение учёта курсантов. Проект реализуется на базе Django и использованием PostgreSQL в качестве СУБД.

Система предназначена для:

* Курсантов, желающих получить дополнительное обучение вождению
* Инструкторов, предлагающих свои услуги

2. Требование к графическому дизайну.

* Интерфейс должен быть адаптивным и доступным на разных устройствах
* Дизайн разрабатывается в Figma с соблюдением принципов UI/UX
* Главная страница содержит краткое описание сервиса, блок для перехода на страницу регистрации.

3. Эксплуатационные требования

* Система должна поддерживать работу на всех современных браузерах
* Установка и развертывание производится на сервере Linux с использованием Nginx
* Проект должен содержать миграции базы данных и README-файл с инструкцией по запуску.

4. Функциональные требования:

Для курсантов: Регистрация и вход в систему, просмотр списка курсов, онлайн-запись на занятие, взаимодействие с курсами, на которые записан курсант.

Для инструкторов: Регистрация и вход в систему, создание, изменение и удаление курса, просмотр заявок от курсантов.

Для администратора: Панель управления пользователями, модерация курсов

5. Требования к надежности и нагрузке: Все формы должны быть защищены от SQL инъекций и XSS-атак, пароли хранятся в хешированном виде (Django auth)

6. Структура проекта.

Проект реализуется по архитектуре MVC на фреймворке Django:

* Models.py – модели пользователей, курсов
* Views.py – контроллеры, реализующие бизнес-логику
* Templates – HTML-шаблоны, основанные на дизайне из Figma
* Static – CSS и медиа файлы
* Forms.py – формы регистрации, авторизации, создание и изменение курсов
* Admin.py – административный интерфейс

Приложения:

* main – основная часть проекта, где реализована логика взаимодействия с курсами
* landing – главная страница для привлечения пользователей

Разработанное техническое задание отражает ключевые аспекты и особенности веб системы «Автокурсы». Учет требований к надежности, масштабируемости и безопасности системы позволяет обеспечить стабильную и удобную работу платформы в условиях реальной нагрузки. Таким образом, разработанное техническое задание послужит прочной основой для дальнейшего этапа, а именно проектирование структуры базы данных и реализации программной логики.

## **1.3 Проектирование архитектуры приложения (MTV)**

Архитектура программного обеспечения – это структурное описание системы, определяющее, из каких компонентов она состоит, как эти компоненты взаимодействуют между собой, и какие принципы используются при проектировании [5].

Основными целями разработки архитектуры являются разделение ответственности между компонентами, обеспечение масштабируемости и модульности, упрощение сопровождения и повышение гибкости системы.

Существуют различные шаблоны архитектуры, например монолитная, клиент-серверная, микросервисная, событийно-ориентированная и так далее. Однако одна из наиболее популярных шаблонов в веб-разработке остаётся MVC(Model-View-Controller) [6].

При разработке веб-приложения «Автокурсы» была выбрана архитектурная модель MTV. Данная модель является реализацией архитектуры MVC с небольшими изменениями под фреймворк Django.

Архитектурные компоненты MTV в Django распределяются следующим образом:

* Model (Модель) – отвечает за работу с базой данных, в этом компоненте определяются структуры таблиц и связи между ними с помощью ORM. В контексте проекта «Автокурсы» модели описывают пользователей, курсы и заявки на обучение.
* Template (Шаблон) – отвечает за отображение данных и визуальную часть интерфейса. Шаблоны в Django используют собственный шаблонизатор, позволяющий динамически формировать HTML на основе переданных данных.
* View (Представление) – служит связующим звеном между моделью и шаблоном. Представление обрабатывает пользовательские запросы, извлекает необходимые данные и передаёт их в шаблон. Именно этот компонент реализует бизнес-логику приложения: обработку регистрации, отображение курсов, запись на курсы, подтверждение заявок.

Подробно изучив архитектурную модель, была разработана структура веб-платформы «Автокурсы» (см. рисунок 1.1).

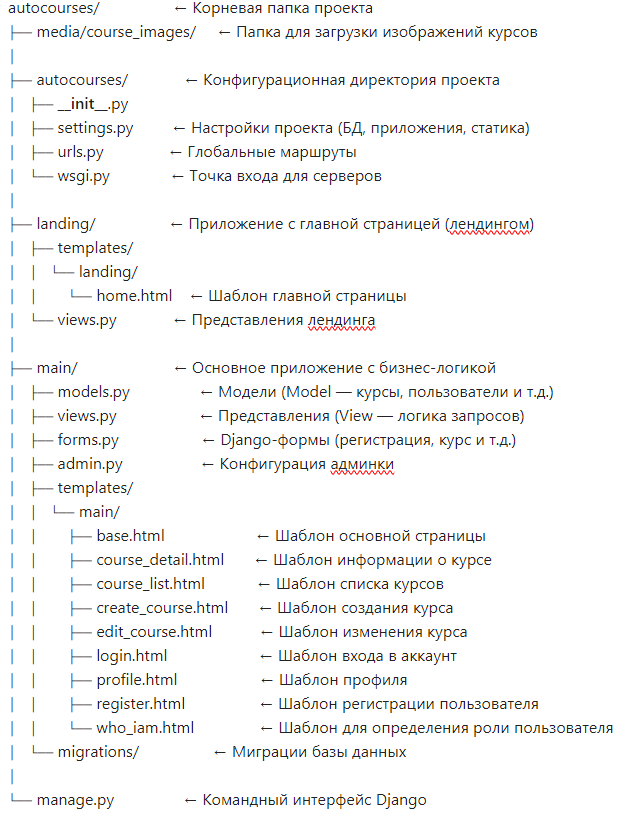


Рисунок 1.1 – Архитектура веб-приложения «Автокурсы»

Таким образом, архитектура MTV является одной из самых популярных и простых моделей, которая позволяет легко организовать структуру проекта по принципам «чистой архитектуры», разделив бизнес-логику, представления и данные.

# **2. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ НА DJANGO**

## **2.1 Структура проекта и настройка среды разработки**

Для реализации веб-ориентированной системы «Автокурсы» была выбрана среда разработки Django, которая предоставляет мощные инструменты для создания полнофункциональных и безопасных сайтов. Django реализует архитектурный шаблон MTV, который обеспечивает разделение логики данных, интерфейса и обработки запросов, что способствует упрощению разработки, тестирования и сопровождения проекта.

Кроме выбора фреймворка важным этапом разработки является выбор инструментов и технологий для работы с сайтом. Таким образом, для реализации проекта использовались следующие инструменты:

* Язык программирования Python 3.13 – удобный и быстрый язык для быстрого прототипирования, обладает широкой экосистемой и активно поддерживается.
* Веб-фреймворк Django – обеспечивает удобную работу с БД, авторизацию и маршрутизацию, а также панель администратора.
* СУБД: PostgreSQL – реляционная СУБД с высокой надёжностью, расширяемостью и поддержкой транзакций.
* Среда разработки: PyCharm – интегрированная среда разработки с поддержкой Django и инструментами отладки.
* Система контроля версий: Git – обеспечивает контроль изменений кода и возможность коллективной разработки.
* UI-дизайн: Figma – используется для проектирования пользовательского интерфейса.

Стандартная организация Django-проекта предусматривает разделение по приложениям, каждое из которых отвечает за отдельную функциональность. Таким образом, в рамках проекта «Автокурсы» были реализованы следующие модули:

* Autocourses – корневая директория, содержащая конфигурационные файлы проекта, а именно настройки, маршрутизация.
* Main – приложение, отвечающее за основную бизнес-логику, а именно отображение курсов, профилей, аутентификацию.
* Landing – приложение, реализующее публичную стартовую страницу с информацией о проекте.

Этот подход обеспечивает модульность и масштабируемость, позволяя при необходимости расширять проект новыми функциональными блоками.

Важным этапом для реализации проекта является корректная настройка среды разработки. Это необходимо для обеспечения одинаковой работы проекта в разных окружениях таких как разработка, тестирование и эксплуатация. К основным этапам настройки относятся:

* Установка всех зависимостей через requirements.txt
* Создание виртуального окружения для изоляции библиотек
* Настройка конфигурационных переменных
* Подключение к локальной базе данных
* Запуск миграций и сборка статики.

Такая организация процесса позволяет обеспечить единообразие окружения, автоматизировать развертывание и снизить вероятность ошибок при запуске проекта на новых машинах. Подробнее про развёртывание веб-приложения будет рассмотрено далее.

Таким образом, на этапе проектирования и настройки среды разработки был заложен прочный фундамент для устойчивой и масштабируемой работы информационной системы. Выбор проверенного стека технологий, а именно Python, Django и PostgreSQL, позволил обеспечить надёжность, безопасность и гибкость проекта. Модульная структура Django-проекта обеспечивает удобство в сопровождении и расширении функциональности, а правильно настроенная среда разработки гарантирует идентичную работу системы в различных окружениях, что критически важно для дальнейших этапов разработки, тестирования и эксплуатации.

## **2.2 Реализация серверной части**

Сервер – это программный компонент устройства, который отвечает за выполнение сервисных функций по запросу клиента. Он представляет доступ к тем или иным ресурсам, возможностям и услугам [7].

Серверная часть представляет собой программу, которая обеспечивает работу сайта. Она отвечает за обработку бизнес-логики, взаимодействие с базой данных и формирование ответов на запросы клиента. В рамках проекта на Django и архитектуры MTV ключевую роль в логике серверной стороны играют модели (Model) и представления (View).

Итак, рассмотрим основные компоненты серверной части:

1. Модели. Они описывают структуру данных, определяют таблицы баз данных и связи между ними. Django используют собственную ORM, что позволяет работать с базой данных через Python-классы. Пример описания модели можно увидеть далее (см. Листинг 2.1).

Листинг 2.1 – Модель пользователя

class User(AbstractUser):

class Role(models.TextChoices):

STUDENT = 'student', 'Студент'

INSTRUCTOR = 'instructor', 'Инструктор'

phone\_number = models.CharField(max\_length=15, blank=True, null=True)

role = models.CharField(max\_length=20, choices=Role.choices)

full\_name = models.CharField(max\_length=255)

birth\_date = models.DateField(null=False, blank=True, default='2000-01-01')

driving\_experience = models.PositiveIntegerField(help\_text='Водительский стаж в годах', default=0)

def \_\_str\_\_(self):

return self.full\_name

Для наглядного представления структуры данных используется ER-диаграмма (см. Рисунок 2.1). Она демонстрирует ключевые сущности системы и взаимодействия между ними. Основными сущностями в приложении являются пользователь, курс и запись на курс. Пользователь может иметь роль студента или инструктора, а также участвовать в создании или прохождении курсов в зависимости от роли. Каждый курс может быть связан с несколькими авторами и содержит основную информацию: название, описание, стоимость, длительность, скрыт ли курс. Модель записи на курс связывает студента и курс, фиксируя статус и дату регистрации. Такая схема обеспечивает наглядное представление логической структуры данных и помогает обеспечить взаимосвязи между объектами системы.

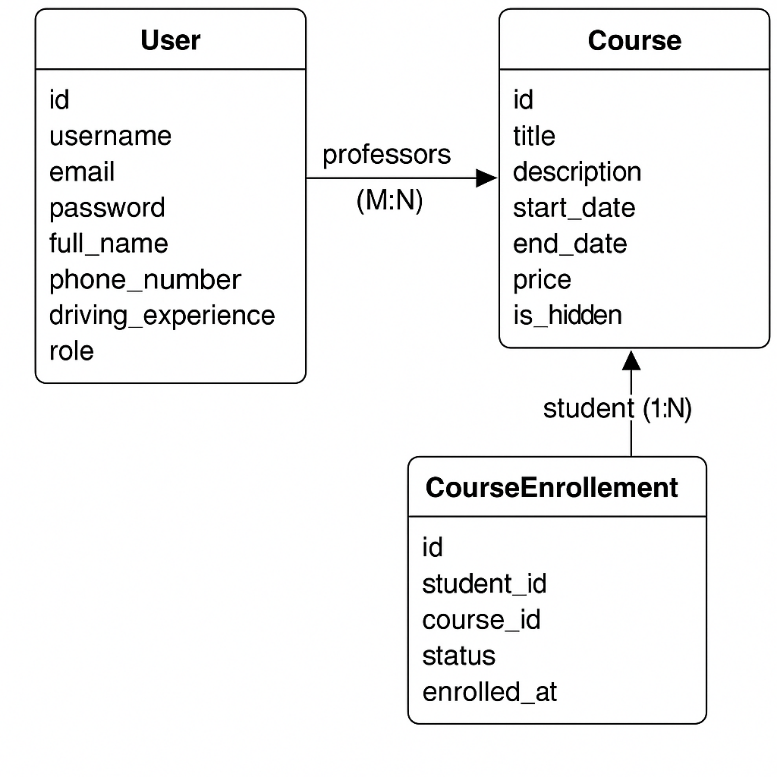


Рисунок 2.1 – ER-диаграмма веб-приложения «Автокурсы»

1. Представления. Этот компонент позволяет обработать входящие запросы, извлекают и модифицируют данные через модели и возвращают соответствующие ответы. Для описания представлений могут использоваться как функции, так и классы Python. Пример создания представлений указан ниже (см. Листинг 2.2)

Листинг 2.2 – Представления создания курса и записи на курс

1. Формы. Django позволяет создавать свои HTML-формы через встроенный механизм, который обеспечивает генерацию полей, валидацию данных, связывание их с моделями. Описание формы создания курса можно увидеть далее (см. Листинг 2.3).

# Представление создания курса

@login\_required

def create\_course(request):

if request.user.role != User.Role.INSTRUCTOR:

return HttpResponseForbidden('Только инструкторы могут создавать курсы.')

if request.method == 'POST':

form = CourseCreationForm(request.POST, request.FILES)

if form.is\_valid():

course = form.save()

course.authors.add(request.user)

messages.success(request, 'Курс успешно создан!')

return redirect(f'/main/courses/{course.id}')

else:

form = CourseCreationForm()

return render(request, 'main/create\_course.html', {'form': form})

# Представление записи на курс

@login\_required

def enroll\_in\_course(request, course\_id):

course = get\_object\_or\_404(Course, id=course\_id)

if request.user.role != User.Role.STUDENT:

messages.error(request, 'Только студенты могут записываться на курсы.')

return redirect(f'/main/courses/{course.id}')

enrollment, created = CourseEnrollment.objects.get\_or\_create(student=request.user, course=course, defaults={'status': 'pending'})

if created:

messages.success(request, 'Заявка на курс отправлена.')

else:

messages.info(request, 'Вы уже подали заявку или записаны на этот курс.')

return redirect(f'/main/courses/{course.id}')

Листинг 2.3 – Форма создания курса

class CourseCreationForm(forms.ModelForm):

class Meta:

model = Course

fields = ('title', 'description', 'start\_date', 'end\_date', 'price', 'is\_hidden', 'image')

widgets = {

'title': forms.TextInput(attrs={'class': 'form-control'}),

'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 4}),

'start\_date': forms.DateInput(attrs={'type': 'date', 'class': 'form-control'}),

'end\_date': forms.DateInput(attrs={'type': 'date', 'class': 'form-control'}),

'price': forms.NumberInput(attrs={'class': 'form-control'}),

'is\_hidden': forms.CheckboxInput(attrs={'class': 'form-check-input'}),

}

1. Административный интерфейс. Django имеет встроенный интерфейс для взаимодействия с данными без необходимости писать отдельные интерфейсы. Это удобно во время разработки, а также для модерации курсов и пользователей. Она создаётся автоматически, однако внешние модели нужно регистрировать (см. Листинг 2.4).

Листинг 2.4 – Регистрация модели курса в административном интерфейсе

@admin.register(Course)

class CourseAdmin(admin.ModelAdmin):

list\_display = ('title', 'price', 'start\_date', 'end\_date', 'image')

search\_fields = ('title',)

filter\_horizontal = ('authors',)

list\_filter = ('start\_date', 'end\_date')

def formfield\_for\_dbfield(self, db\_field, request, \*\*kwargs):

if db\_field.name == 'image':

kwargs['widget'] = admin.widgets.AdminFileWidget()

return super().formfield\_for\_dbfield(db\_field, request, \*\*kwargs)

1. URL-маршрутизация. Каждое представление привязывается к определённому URL-шаблону, обеспечивая структуру навигации в приложении. Навигация в приложении «Автокурсы» осуществляется по следующему паттерну (см. Листинг 2.5).

Листинг 2.5 – URL-маршрутизация приложения «Автокурсы»

urlpatterns = [

path("courses/", views.course\_list, name="course\_list"),

path("courses/<int:course\_id>/", views.course\_detail, name="course\_detail"),

path("courses/<int:course\_id>/enroll/", views.enroll\_in\_course, name="enroll\_in\_course"),

path("courses/<int:course\_id>/unenroll/", views.unenroll\_from\_course, name="unenroll\_from\_course"),

path( "courses/<int:course\_id>/delete\_course/", views.delete\_course, name="delete\_course"),

path("enrollments/<int:enrollment\_id>/confirm/", views.confirm\_enrollment, name="confirm\_enrollment"),

path("profile/", views.profile, name="profile"),

path('register/student/', views.register\_student, name='register\_student'),

path('register/instructor/', views.register\_instructor, name='register\_instructor'),

path("login/", views.CustomLoginView.as\_view(template\_name="main/login.html", authentication\_form=CustomAuthenticationForm), name="login"),

path('register', views.register, name='register'),

path("logout/", auth\_views.LogoutView.as\_view(next\_page="/main/login"), name="logout"),

path("courses/create/", views.create\_course, name="create\_course"),

path('reject\_enrollment/<int:enrollment\_id>/', views.reject\_enrollment, name='reject\_enrollment')

]

Таким образом, была реализована полноценная серверная часть приложения с использованием фреймворка Django. Модели описывают ключевые сущности системы: курсы, пользователи, отношения между ними. Через Django ORM обеспечивается безопасный и удобный доступ к данным. Для обработки пользовательского ввода созданы формы, отвечающие за регистрацию, аутентификацию и управление курсами. Представления обеспечивают устойчивую работу с URL-маршрутами и обработку GET и POST-запросов. Также была настроена маршрутизация URL`ов, обеспечивающая логичную структуру адресов и надёжную навигацию между страницами.

Все эти технические решения обеспечили отказоустойчивость, модульность и масштабируемость серверной части системы «Автокурсы».

## **2.3 Реализация клиентской части**

Клиентская часть веб-приложения отвечает за отображение данных пользователю и взаимодействие с серверной логикой через HTTP-запросы. В Django за клиентскую сторону отвечает шаблонизатор DTL (Django Template Language), который позволяет внедрять данные из представлений в HTML-страницы.

Благодаря такому подходу можно выделить несколько основных преимуществ – поддержка базовых конструкций управления, а именно циклы и условия, экранирование данных по умолчанию (позволяет реализовать XSS-защиту), а также простое расширение через теги и фильтры.

Для стилизации в проекте используется CSS и CSS-фреймворк Bootstrap.

Веб-приложение «Автокурсы» разделено на 2 приложения. Первое приложение – main – отвечает за реализацию страниц регистрации, авторизации, профиля и взаимодействий с курсами. Второе приложение отвечает за главную страницу, призванную погрузить пользователя в продукт «Автокурсы». Структура шаблонов описана ниже (см. Рисунок 2.2)

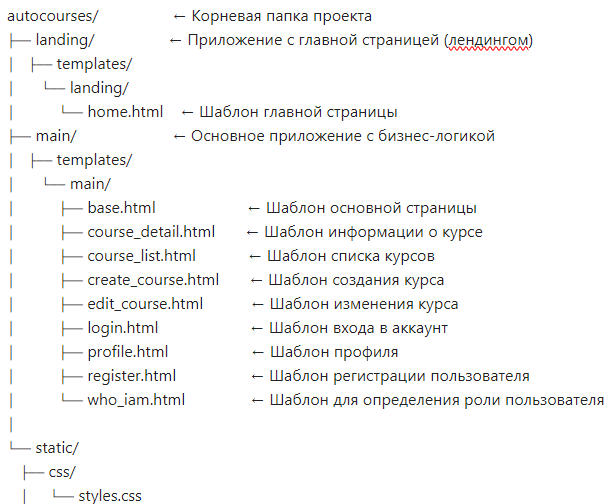
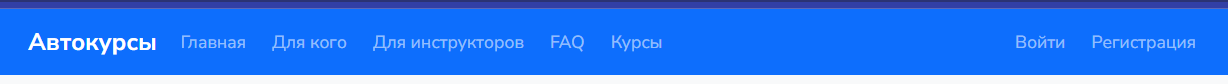


Рисунок 2.2 – Структура шаблонов в проекте «Автокурсы»

Реализация и взаимодействие шаблонов строится на основе иерархии – большинство страниц наследуются от base.html (см. Листинг 2.6), что обеспечивает единый стиль интерфейса и повторное использование компонентов, таких как навигация, подвал сайта (см. Рисунок 2.3).

Листинг 2.6 – Реализация шаблона base.html 

{% load static %}

<!DOCTYPE html>

<html lang="ru">

<head>

<title>{% block title %}Автокурсы{% endblock %}</title>

<link rel="stylesheet" href="{% static 'css/styles.css' %}" />

</head>

<body>

<nav class="navbar navbar-expand-lg sticky-top">

<div class="container">

<a href="{% url 'landing:home' %}">Автокурсы</a>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav me-auto">

<li class="nav-item">

<a href="{% url 'main:course\_list' %}">Курсы</a>

</li>

</ul>

<ul class="navbar-nav">

{% if user.is\_authenticated %}

<li class="nav-item">

<a class="nav-link" href="{% url 'main:profile' %}">

{{ user.full\_name }}

</a>

</li>

{% else %}

<li class="nav-item">

<a href="{% url 'main:login' %}">Войти</a>

</li>

{% endif %}

</ul>

</div>

</div>

</nav>

<main>

{% block content %}{% endblock %}

</main>

</body>

</html>

Рисунок 2.3 – Интерфейс шаблона base.html

В шаблонах используются встроенные теги for, if, url, csrf\_token и передача переменных через контекст для отображения курсов (см. Листинг 2.7 и Рисунок 2.4).

Листинг 2.7 – Реализация шаблона course\_list.html

{% extends 'main/base.html' %}

{% block title %}Список курсов{% endblock %}

{% block content %}

<div class="container py-5">

<h1 class="mb-5 text-center">Список курсов</h1>

<div class="row g-4">

{% for course in courses %}

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

<div class="card course-card shadow-sm h-100">

{% if course.image %}

<img src="{{ course.image.url }}" class="card-img-top course-image" alt="{{ course.title }}">

{% else %}

<div class="course-image-placeholder d-flex align-items-center justify-content-center">

<i class="bi bi-image fs-1 text-secondary"></i>

</div>

{% endif %}

<div class="card-body d-flex flex-column">

<h5 class="card-title">{{ course.title }}</h5>

<p class="card-text flex-grow-1 text-secondary line-clamp-5">{{ course.description|linebreaksbr}}</p>

<p class="card-text flex-grow-1 text-secondary">

<strong>Дата начала:</strong> {{ course.start\_date }}<br>

<strong>Дата окончания:</strong> {{ course.end\_date }}<br>

<strong>Стоимость:</strong> {{ course.price }} руб.

</p>

<a href="{% url 'main:course\_detail' course.id %}" class="btn btn-primary mt-3">Подробнее</a>

</div>

</div>

</div>

{% empty %}

<div class="col-12 text-center">

<p class="text-muted">Нет доступных курсов.</p>

</div>

{% endfor %}

</div>

</div>

{% endblock %}

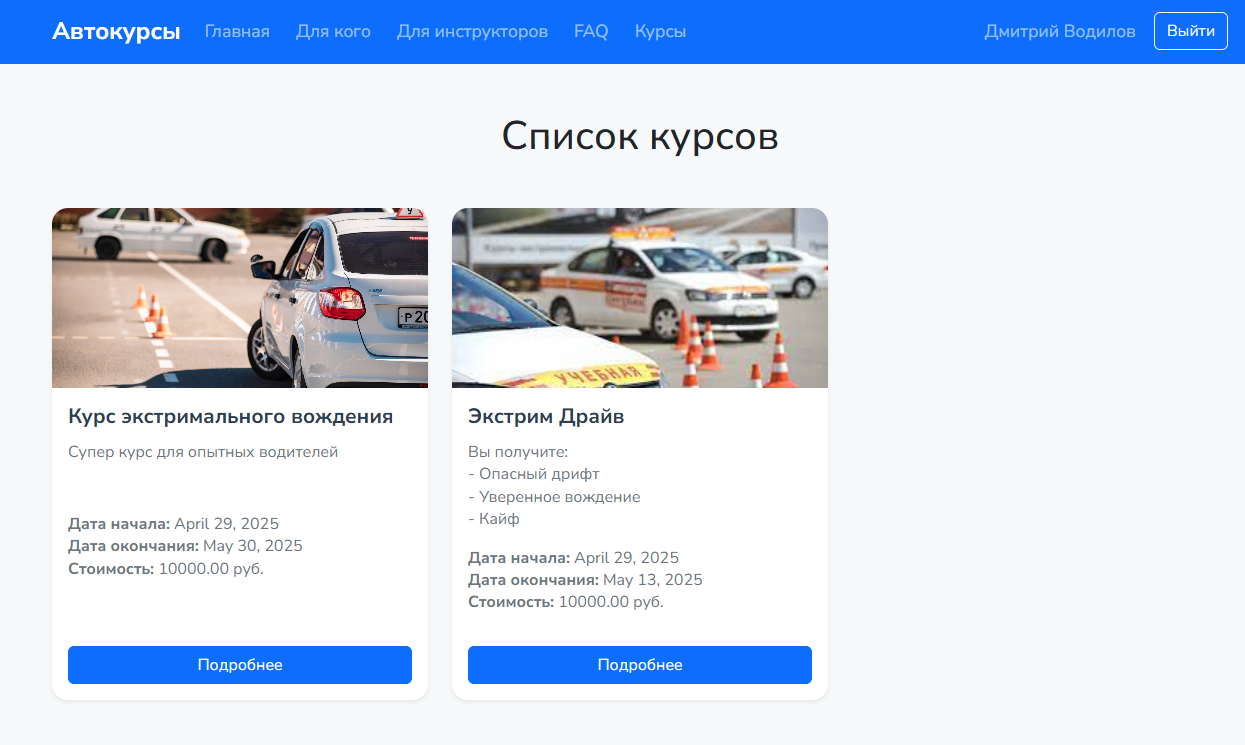


Рисунок 2.4 – Интерфейс шаблона course\_list.html

Формы, созданные в forms.py, рендерятся с помощью шаблонных тегах и обрабатываются в представлениях (см. Листинг 2.8 и Рисунок 2.5).

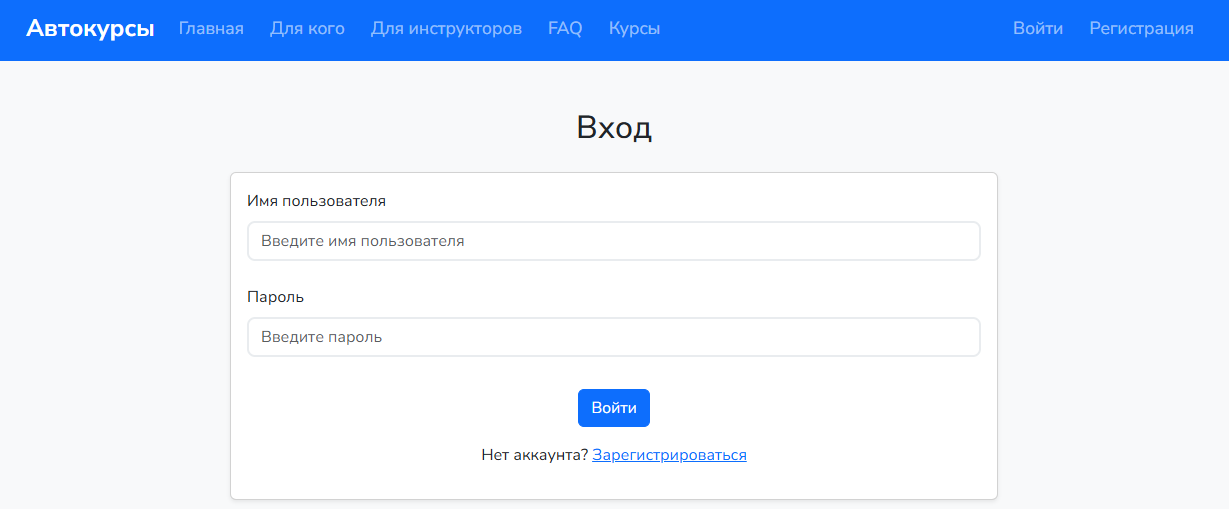


Рисунок 2.5 – Интерфейс шаблона login.html

Листинг 2.8 – Реализация шаблона login.html

{% extends 'main/base.html' %}

{% block title %}Вход{% endblock %}

{% block content %}

<div class="form-card my-5" >

<div class="card shadow-sm">

<form method="post">

{% csrf\_token %}

{% for field in form %}

<div class="form-group">

<label for="{{ field.id\_for\_label }}" class="form-label">{{ field.label }}</label>

{{ field }}

{% if field.errors %}

{% for error in field.errors %}

<div>{{ error }}</div>

{% endfor %}

{% endif %}

</div>

{% endfor %}

{% if form.non\_field\_errors %}

{% for error in form.non\_field\_errors %}

<div class="form-error">{{ error }}</div>

{% endfor %}

{% endif %}

<div class="form-actions">

<button type="submit">Войти</button>

</div>

</form>

</div>

</div>

{% endblock %}

Для визуального оформления используется CSS-стилизация, а также подключены шрифты и сетка, упрощающая адаптивную верстку под разные устройства (см. Листинг 2.9). Чтобы создать CSS классы для шаблонов формы, создающиеся автоматически через Django, в forms.py прописываются отдельные атрибуты class со значениями названия CSS класса. Для перенаправления после регистрации или авторизации в settings.py указываются специальные параметры:

* LOGIN\_REDIRECT\_URL = "/main/courses"
* LOGOUT\_REDIRECT\_URL = "login"

Листинг 2.9 – Стилизация форм из forms.py

.form-card {

max-width: 40%;

margin: 0 auto;

}

.form-group {

margin-bottom: 1.5rem;

}

.form-control {

border-radius: 8px;

border: 2px solid #e9ecef;

transition: border-color 0.3s, box-shadow 0.3s;

}

.form-control:focus {

border-color: #007bff;

box-shadow: 0 0 8px rgba(0, 123, 255, 0.3);

}

.form-error {

color: var(--bs-red);

font-size: 0.875rem;

margin-top: 0.25rem;

}

.form-actions {

text-align: center;

margin-top: 2rem;

}

Таким образом, клиентская часть проекта «Автокурсы» была выполнена с применением шаблонной системы Django, обеспечивающей безопасную и гибкую генерацию HTML-страниц. Благодаря базовому шаблону base.html и механизму наследования удалось централизовать дизайн и повторно использовать элементы интерфейса.

Были реализованы пользовательские интерфейсы для всех ключевых компонентов: регистрация и вход, работа с курсами, страница профиля и списка курсов. Взаимодействие с сервером осуществляется через стандартные HTML-формы и встроенную CSRF-защиту. Вся верстка адаптирована под удобство работы конечного пользователя.

# **3. РАЗВЁРТЫВАНИЕ И ЭКСПЛУАТАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Разработка и внедрение информационной системы включает не только создание функциональной части, но и подготовку к эксплуатации. На этапе развертывания системы и сопровождения проекта важна установка, конфигурации и миграции, которые формируют основу стабильной и корректной работы системы [8]. Рассмотрим эти этапы подробнее.

Для развертывания приложения важно учитывать системные требования к серверу, где будет расположено приложение. Важными компонентами являются:

* Операционная система: Linux (Ubuntu 22.04) или Windows
* Установленные компоненты: Python 3.13, PostgeSQL, pip, virtualenv
* Обязательные сервисы: Nginx, Gunicorn

Перед дальнейшей работой предварительно создаётся виртуальное окружение. Для этого нужно выполнить следующие шаги:

* Python -m venv .venv
* Для запуска в Linux: source .venv/bin/activate
* Для запуска в Windows: .venv\Scripts\activate

1. Установка – это процесс переноса программного обеспечения из среды разработки в рабочую среду. Этот этап включает в себя установку всех необходимых компонентов и зависимостей, таких как интерпретаторы языков программирования, библиотеки, фреймворки и базы данных. Для корректной работы проекта на целевом сервере необходимо установить зависимости, указанные в файле requirements.txt в папке проекта.
   * Pip install -r requirments.txt
2. Настройка конфигурации. Конфигурация – это настройка параметров системы в соответствии с конкретными требованиями окружения. Под конфигурацией понимается задание переменных, определение путей к файлам, настройка параметров безопасности, логирования и так далее. Цель настройки конфигурации – адаптировать систему к условиям её использования.

Настройка проекта разделяются на локальные и продакшн-конфигурации. Основные параметры:

* DEBUG =False – отключение режима отладки.
* ALLOWED\_HOSTS = [‘домен.сайта’, ‘ip-адрес’]
* Пути к статическим и медиафайлам (см. Листинг 3.1)

Листинг 3.1 – Путь к статическим и медиафайлам

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static/')

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media/')

* Настройка подключения к базе данных (см. Листинг 3.2)

Листинг 3.2 – Настройка подключения к БД PostgreSQL

DATABASES = {

"default": {

"ENGINE": "django.db.backends.postgresql",

"NAME": "autocourses\_db",

"USER": "postgres",

"PASSWORD": "qwer123",

"HOST": "localhost",

"PORT": "5432",

}

}

1. Миграции – это механизм внесения изменений в структуру базы данных в соответствии с изменениями, произведёнными в логической модели данных приложения. Этот механизм позволяет безопасно и последовательно обновлять схему базы данных без необходимости вручную писать SQL-запросы.

Для выполнения миграций используются следующие команды (см. Листинг 3.3):

Листинг 3.3 – Команды для создания миграций

python manage.py makemigrations

python manage.py migrate

1. Сборка статических файлов. Статические файлы необходимо собрать в одну директорию с помощью команды python manage.py collectstatic. Это важно для дальнейшей раздачи через веб-сервер, например, Nginx.

После выполнения всех инструкций настройки и развертывания серверного программного обеспечения переходим к этапу сопровождения проекта. Этот этап включает в себя:

* Регулярное обновление зависимостей и применение патчей безопасности
* Мониторинг логов через nginx.access.log
* Плановое резервное копирование базы данных
* Контроль за производительность и своевременное обновление сервера.
* Обработка обращений пользователей и устранение ошибок.

Таким образом, правильная инструкция по развёртыванию и эксплуатации проекта является критически важным условием для его стабильной работы. Она упрощает передачу проекта другим разработчикам, ускоряет процесс развёртывания и снижает риски, связанные с человеческим фактором. На этапе установки и конфигурации выполняется подготовка среды исполнения для стабильной работы веб-приложения, корректно выполненные миграции создают структуру базы данных, а сборка статики позволяет обеспечить правильную работоспособность клиентской части.

# **ЗАКЛЮЧЕНИЕ**

Целью данной курсовой работы являлась разработка веб-ориентированной информационной системы «Автокурсы». Данная система предназначена для автоматизации процессов управления курсами и взаимодействия между учениками и инструкторами в едином веб-приложении. В процессе выполнения курсовой работы поставленная цель была успешно достигнута.

Для достижения цели были выполнены следующие поставленные задачи:

1. Проведён анализ исследуемой области, на базе которой сформированы функциональные и технические требования к веб-приложению
2. Составлена архитектура проекта, включающая структуру базы данных и модель взаимодействия компонентов.
3. Реализована серверная часть системы с использованием фреймворка Django, который обеспечивает надёжную и легко масштабируемую основу для дальнейшего использования.
4. Создан современный пользовательский интерфейс, отвечающий всем UI/UX требованиям, адаптированный под различные устройства, с использованием HTML, CSS, JavaScript и Bootstrap.
5. Выполнено развёртывание проекта и настройка среды на сервере, что обеспечивает стабильную работу информационной системы.

Актуальность разработанной системы обусловлена важной и старой проблемой: современные автошколы зачастую ориентированы только на минимальное соответствие требованиям для получения водительского удостоверения, при этом не обеспечивая достаточной подготовки к реальному дорожному движению. Это приводит к тому, что многие выпускники автошкол чувствуют себя неуверенно на дорогах, что может привести к опасным последствиям. В связи с этой проблемой наблюдается рост спроса на специализированные курсы по вождению в реальных условиях. Таким образом, разрабатываемая система «Автокурсы» была призвана решить данную проблему, предоставляя удобный инструмент для поиска, оценки и записи на различные курсы по вождению.

В ходе выполнения проекта были использованы современные технологии, а именно Python 3.13, Django, PostgreSQL, систему контроля версий Git, графический редактор Figma и интегрированную среду разработки PyCharm. Данный выбор позволил добиться высокой степени надёжности, расширяемости и удобства в сопровождении системы.

С учётом развития современных информационных технологий веб-ориентированная информационная система «Автокурсы» имеет большой потенциал для дальнейшего масштабирования – например, за счёт интеграции с онлайн-платёжными системами, мобильными платформами и расширением функционала по управлению обучающими модулями.

Таким образом, в ходе выполнения курсовой работы была спроектирована и реализована современная, надёжная и масштабируемая веб-ориентированная система, которая отвечает актуальным требованиям пользователей и рынка образовательных услуг. Использование проверенных технологий, продуманной архитектуры и фокус на реальных потребностях целевой аудитории позволили создать инструмент, который способен существенно повысить качество и доступность курсов вождения. Проект подтверждает эффективность комплексного подхода к разработке информационных систем и демонстрирует готовность к внедрению в реальную эксплуатацию с перспективой дальнейшего развития.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Абрамов, С. М. PostgreSQL. Подробное руководство / С. М. Абрамов. – Москва: ДМК Пресс, 2021. – 944 с.

2. Васильев, А. Н. Django. Разработка веб-приложений на Python / А. Н. Васильев. – Москва: Наука и техника, 2022. – 416 с.

3. Новые водители: чему их учат, сколько это стоит и почему снова критикуют автошколы // За рулём URL: https://www.zr.ru/content/articles/967906-novye-voditeli-chemu-uchat-sko/ (дата обращения: 04.05.2025).

4. Как составить техническое задание и получить то, что нужно // Контур URL: https://kontur.ru/articles/5945 (дата обращения: 04.05.2025).

5. Архитектура программного обеспечения // КОД URL: https://thecode.media/arhitektura-programmnogo-obespecheniya/ (дата обращения: 04.05.2025).

6. Архитектура в Django проектах — как выжить // Хабр URL: https://habr.com/ru/companies/vivid\_money/articles/544856/ (дата обращения: 04.05.2025).

7. Серверная часть // Appmaster URL: https://appmaster.io/ru/glossary/servernaia-chast-ru (дата обращения: 04.05.2025).

8. Развертывание Django-приложений // skypro URL: https://sky.pro/wiki/python/razvertyvanie-django-prilozhenij/ (дата обращения: 04.05.2025).

# **Приложение А. Листинг исходных кодов**

Ввиду большого объема исходных текстов, проект задания располагается на облачном ресурсе GitHub по ссылке: https://github.com/student/project и прикладывается на электронном носителе к настоящей пояснительной записке.