



МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ –
МСХА имени К.А. ТИМИРЯЗЕВА»
(ФГБОУ ВО РГАУ - МСХА имени К.А. Тимирязева)

Институт экономики и управления АПК
Кафедра статистики и кибернетики

Инженерия информационных систем

КУРСОВОЙ ПРОЕКТ

на тему: Разработка конструктора резюме на базе фреймворка Django

Выполнил
обучающийся 1 курса
группы ДЭ 37-24

Гваськова Лилия Римовна
ФИО

Дата регистрации КП
на кафедре _____

Допущен (а) к защите

Руководитель:

ученая степень, ученое звание, ФИО

Члены комиссии:

ученая степень, ученое звание, ФИО

подпись

ученая степень, ученое звание, ФИО

подпись

ученая степень, ученое звание, ФИО

подпись

Оценка _____

Дата защиты _____

Москва, 2025

Аннотация

Настоящий документ является курсовым проектом по разработке конструктора резюме (далее – конструктор резюме, проект либо Система).

В главе 1 приведен анализ конкурентов и список преимуществ разрабатываемого конструктора резюме.

В главе 2 описана реализация Системы:

- Описание функциональных возможностей Системы (раздел 2.1);
- Описание архитектуры проекта, его структуры, моделей и программных интерфейсов, предоставляемых Системой (раздел 2.2);
- Этапы разработки (раздел 2.3);
- Пользовательские сценарии для тестирования Системы (раздел 2.4).

В приложении приведен код конструктора резюме.

Содержание

Введение	3
Глава 1. Сравнительный анализ	4
1.1 Основные проблемы существующих конструкторов резюме	4
1.2 Потребность в улучшенном решении	5
Глава 2. Реализация Системы	7
2.1 Анализ Системы	7
2.2 Проектирование Системы	9
2.2.1 Описание архитектуры	9
2.2.2 Описание структуры	10
2.2.3 Описание маршрутов	12
2.2.4 Описание моделей	12
2.2.5 Описание интерфейса	15
2.3 Разработка Системы	21
2.3.1 Подготовка среды разработки	21
2.3.2 Создание и регистрация приложения	21
2.3.3 Создание суперпользователя	22
2.3.4 Создание моделей и применение миграций	22
2.3.5 Создание представлений, форм и маршрутов	23
2.3.6 Создание шаблонов	23
2.3.7 Запуск сайта	23
2.4 Тестирование Системы	24
2.4.1 Аутентификация пользователя	24
2.4.2 Создание резюме	24
2.4.3 Просмотр резюме	25
2.4.4 Редактирование резюме	26
2.4.5 Удаление резюме	26
2.4.6 Экспорт резюме в PDF формат	27
Заключение	28
Библиографический список	29
Приложение А	30

Введение

В условиях поиска работы важным моментом для соискателей является возможность упростить процесс создания резюме. Эту задачу решает автоматизация процесса создания резюме с помощью специализированного конструктора, благодаря чему отпадает необходимость ручного составления и форматирования резюме. Данная система особенно актуальна для студентов и начинающих специалистов, так как экономит время и минимизирует ошибки при подготовке резюме.

Целью данного проекта является разработка конструктора резюме, который позволит автоматизировать процесс создания резюме. Для решения данной цели необходимо было решить следующие задачи:

- Изучить фреймворк Django;
- Спроектировать логику работы Системы;
- Написать код для реализации функциональности Системы;
- Описать процесс разработки.

Глава 1. Сравнительный анализ

В условиях высокой конкуренции на рынке труда ключевым инструментом для соискателей становится резюме – документ, который представляет кандидата работодателю. Однако создание качественного резюме требует не только профессиональных навыков, но и внимания к деталям оформления. Именно поэтому онлайн-конструкторы резюме стали популярным решением для тех, кто хочет быстро и эффективно подготовить документ, соответствующий требованиям рынка труда.

Несмотря на широкий выбор доступных конструкторов резюме, многие из них имеют значительные ограничения, которые затрудняют процесс создания качественного документа. Эти ограничения могут проявляться как в технических недостатках (например, сложности с экспортом или печатью), так и в дизайнерских решениях (шаблоны, которые не подходят для определенных профессий или отраслей). В результате пользователи сталкиваются с рядом проблем, которые снижают эффективность использования таких сервисов.

1.1 Основные проблемы существующих конструкторов резюме

Рассмотрим наиболее известные платформы для создания резюме и выделим их ключевые недостатки.

1. HH.ru

Один из крупнейших российских порталов для поиска работы, HH.ru, предлагает встроенный конструктор резюме. Однако его функционал имеет ряд существенных ограничений. Во-первых, оформление резюме при экспорте в PDF часто выглядит перегруженным, что снижает читаемость документа. Во-вторых, шаблоны резюме на платформе являются однотипными и не предоставляют

пользователям гибко управлять дизайном. Наконец, отсутствует адаптация под печать – при сохранении в PDF возможны проблемы с форматированием.

2. CVmaker (cvmkr.com)

CVmaker – международная платформа, которая позиционируется как универсальный инструмент для создания резюме. Однако ее дизайн и функционал также имеют свои минусы. Например, шаблоны резюме на этой платформе выглядят слишком консервативно и не подходят для творческих профессий, таких как дизайнеры, маркетологи или копирайтеры. Кроме того, большинство стилей и функций доступны только в платной версии, что ограничивает возможности пользователей с бесплатной учетной записью.

3. MyResume.ru

Еще один популярный конструктор резюме, MyResume.ru, предлагает гибкие настройки для создания документа. Однако здесь возникает другая проблема: интерфейс платформы перегружен советами и подсказками, что может отвлекать пользователя от основной задачи – быстрого заполнения резюме. Кроме того, несмотря на наличие множества опций для редактирования, процесс настройки может быть достаточно сложным и затратным по времени. Это особенно заметно для начинающих пользователей, которым требуется простой и понятный инструмент.

1.2 Потребность в улучшенном решении

Учитывая вышеописанные проблемы, становится очевидным, что имеющиеся конструкторы резюме не полностью удовлетворяют потребности пользователей. Современный рынок требует разработки нового решения, которое будет сочетать в себе удобство использования, гибкость в настройке и высокое качество экспорта.

Основными преимуществами разрабатываемого решения являются:

- Интуитивно понятный интерфейс, который позволит пользователям быстро освоить платформу и сосредоточиться на содержании резюме, а не на его оформлении.
- Оптимизация процесса создания резюме, включая использование стандартных блоков, которые помогают структурировать информацию и экономить время.
- Высокое качество экспорта, обеспечивающее профессиональный внешний вид резюме при сохранении в формате PDF.

Таким образом, разрабатываемый конструктор резюме может стать конкурентоспособной альтернативой существующим платформам.

Глава 2. Реализация Системы

Реализация Системы ведется в учебно-познавательных целях, как плацдарм для начала освоения программирования: изучение объектно-ориентированного подхода при разработке программного обеспечения, а также изучение UML языка и построение диаграмм с помощью программного обеспечения draw.io.

2.1 Анализ Системы

На этапе формирования общих требований к Системе была построена диаграмма вариантов использования (рисунок 1), которая описана в [1, с. 29] и [2, с. 91]. Список функциональных групп (ролей) пользователей Системы:

- Пользователь;
- Администратор.

Ниже приведен список функциональных возможностей Системы.

Пользователь может:

- Зарегистрироваться в системе;
- Аутентифицироваться в системе;
- Создать резюме;
- Добавить/удалить раздел резюме;
- Выбрать шаблон резюме;
- Редактировать резюме;
- Удалить резюме;
- Посмотреть резюме;
- Импортировать резюме (PDF, Word);
- Экспортировать резюме (PDF, Word);
- Проверить правописание;
- Улучшить резюме;

- Сгенерировать сопроводительное письмо;
- Получить рекомендацию.

Администратор может:

- Управлять шаблонами резюме;
- Управлять пользователями.

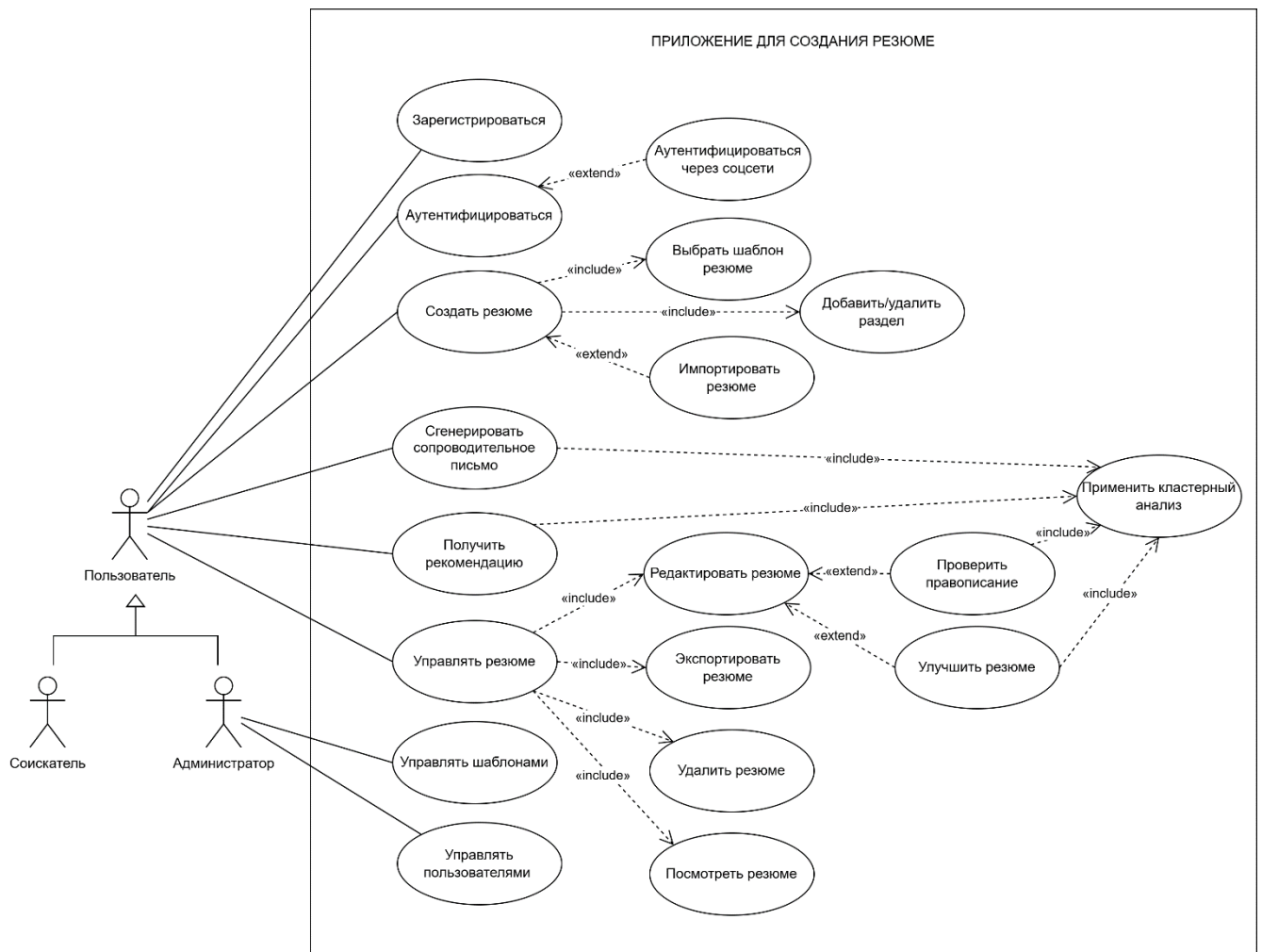


Рисунок 1 – Диаграмма вариантов использования

Диаграмма вариантов использования наглядно демонстрирует взаимодействие пользователей с системой и помогает определить ключевые сценарии использования, которые необходимо реализовать для обеспечения функциональности приложения.

2.2 Проектирование Системы

2.2.1 Описание архитектуры

Конструктор резюме – это веб-приложение, предназначенное для создания и управления резюме. Система построена по клиент-серверной архитектуре (рисунок 2) и состоит из следующих компонентов:

1. Панель администратора представляет собой готовый веб-интерфейс для управления данными, доступный только для пользователей с правами администратора;
2. Клиентская часть – веб-интерфейс для взаимодействия пользователей с Системой;
3. Серверная часть реализована на базе фреймворка Django [3] и отвечает за обработку запросов и управление данными;
4. База данных (SQLite) используется для хранения информации о пользователях, резюме и разделах резюме.

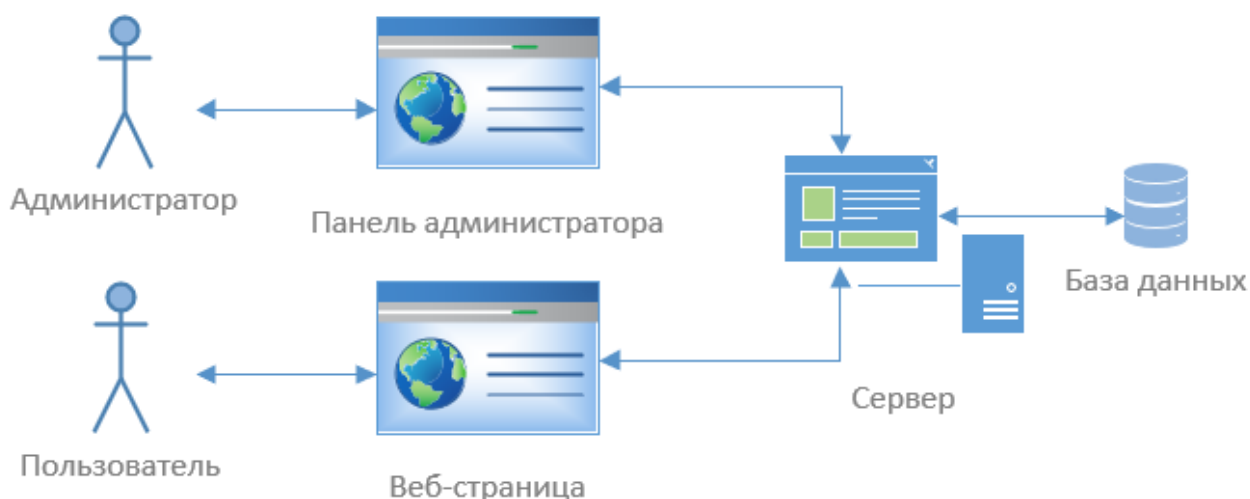


Рисунок 2 – Архитектура Системы

Клиент-серверная архитектура позволяет разделить логику обработки данных и их отображения, что упрощает поддержку и развитие системы.

2.2.2 Описание структуры

На рисунке 3 приведена структура приложения на Django.

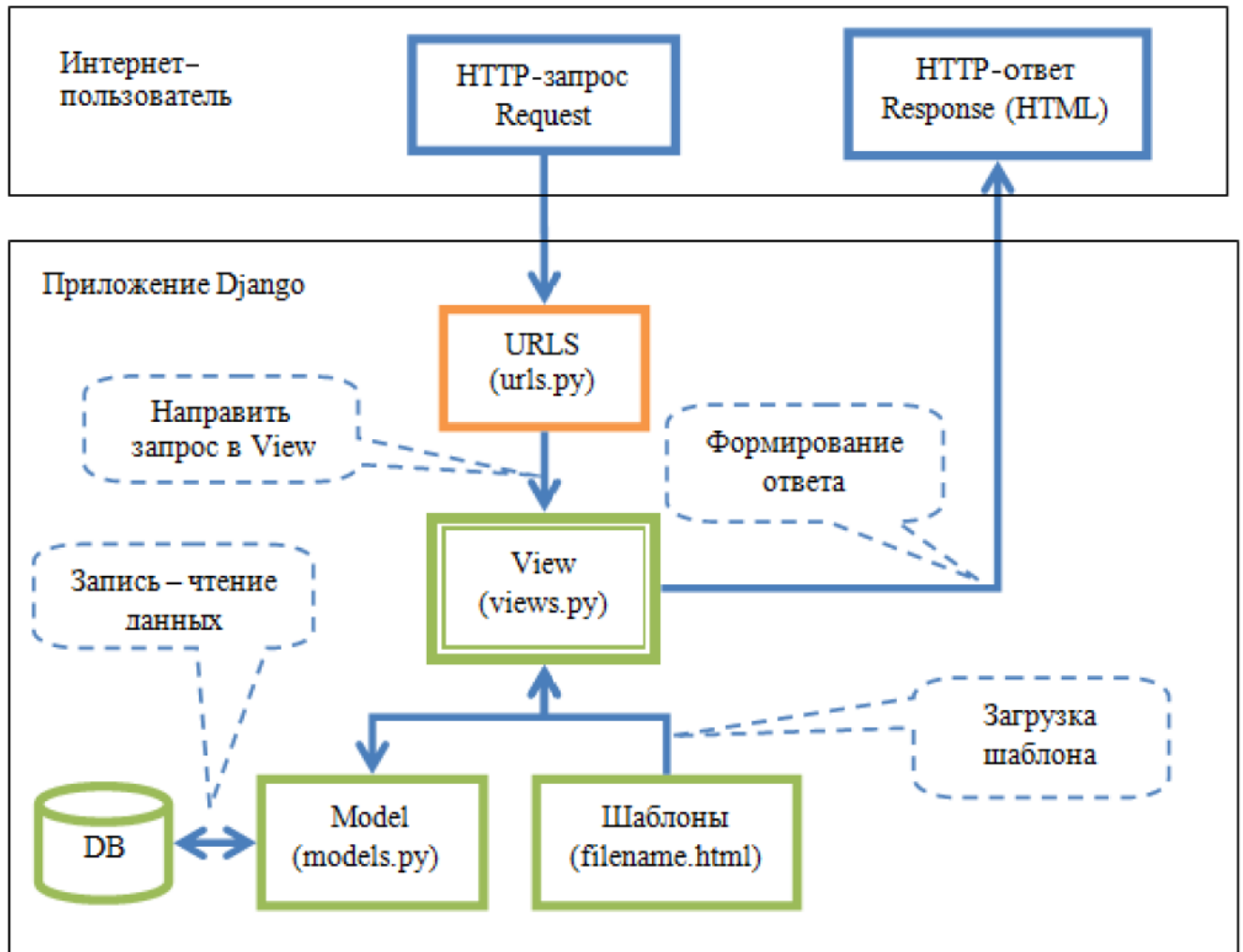


Рисунок 3 – Базовые блоки веб-приложения на Django

Приложение на Django состоит из четырех основных блоков [4, с. 105]:

1. Маршруты (URLs) – определяют, как HTTP-запросы направляются к соответствующим представлениям (View) на основе URL. Данные, извлеченные из URL, передаются в представления в виде аргументов.
2. Модели (Models) – определяют структуру данных приложения и предоставляют методы для работы с базой данных (добавление, изменение, удаление, запросы).

3. Представления (Views) – обрабатывают запросы, взаимодействуют с моделями для работы с данными и возвращают результат (обычно HTML) через шаблоны.

4. Шаблоны (Templates) – текстовые файлы (например, HTML), определяющие структуру страниц. Используются для подстановки данных из базы или ввода пользователя.

Когда к приложению Django приходит запрос от удаленного пользователя (HTTP-запрос), то URL-диспетчер определяет, с каким ресурсом нужно сопоставить этот запрос, и передает его выбранному ресурсу. Ресурсом в этом случае является представление (view), которое, получив запрос, обрабатывает его определенным образом. В процессе обработки запроса представление (view) может обращаться через модели (model) к базе данных (БД), получать из нее данные или, наоборот, сохранять информацию в БД. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет собой сгенерированный HTML-код, для генерации которого применяются шаблоны (template).

Ниже приведена структура проекта «Конструктор резюме» и краткое описание ее элементов.

resume_builder/	# Папка сайта
├── resume_builder/	# Папка проекта
│ ├── __pycache__/	# Кэшированные файлы Python
│ ├── __init__.py	# Файл, указывающий, что это Python-пакет
│ ├── asgi.py	# Точка входа для ASGI-серверов
│ ├── settings.py	# Настройки проекта
│ ├── urls.py	# Ассоциации url адресов с представлениями
│ └── wsgi.py	# Точка входа для WSGI-серверов
├── resumes/	# Папка приложения
│ ├── __pycache__/	# Кэшированные файлы Python
│ ├── migrations/	# Миграции
│ ├── templates/	# Шаблоны (HTML-файлы)
│ ├── __init__.py	# Файл, указывающий, что это Python-пакет
│ ├── admin.py	# Регистрация моделей в админке
│ ├── apps.py	# Конфигурация приложения
│ ├── forms.py	# HTML-формы
│ ├── models.py	# Модели
│ ├── tests.py	# Тесты
│ ├── urls.py	# Маршруты
│ └── views.py	# Представления
├── db.sqlite3	# База данных
└── manage.py	# Скрипт для управления проектом

2.2.3 Описание маршрутов

В таблице 1 приведен перечень URL-адресов, которые понадобятся для конструктора резюме.

Таблица 1 – URL-адреса конструктора резюме

URL-адрес	Описание
/resumes/login/	Страница аутентификации
/resumes/logout/	URL для выхода из системы
/resumes/my-resumes/	Список всех резюме пользователя
/resumes/resume/new/	Страница создания резюме
/resumes/resume/<id>/	Страница просмотра резюме
/resumes/resume/<id>/edit/	Страница редактирования резюме
/resumes/resume/<id>/delete/	Страница удаления резюме
resume/<id>/export-pdf/	URL для экспорта резюме в формат PDF

Для идентификации конкретного резюме используется значение первичного ключа <id>. Например, URL-адрес /resumes/resume/3/ указывает на резюме с идентификатором 3.

2.2.4 Описание моделей

Для конструктора резюме моделями являются пользователи, резюме и отдельные разделы резюме:

- Персональные данные;
- Образование;
- Опыт работы;
- Навыки.

Модели, их характеристики (атрибуты, операции), зависимости между моделями, включая их множители, отображены на диаграмме классов (рисунок 4), которая описана в [5, с. 62] и [6, с. 42].

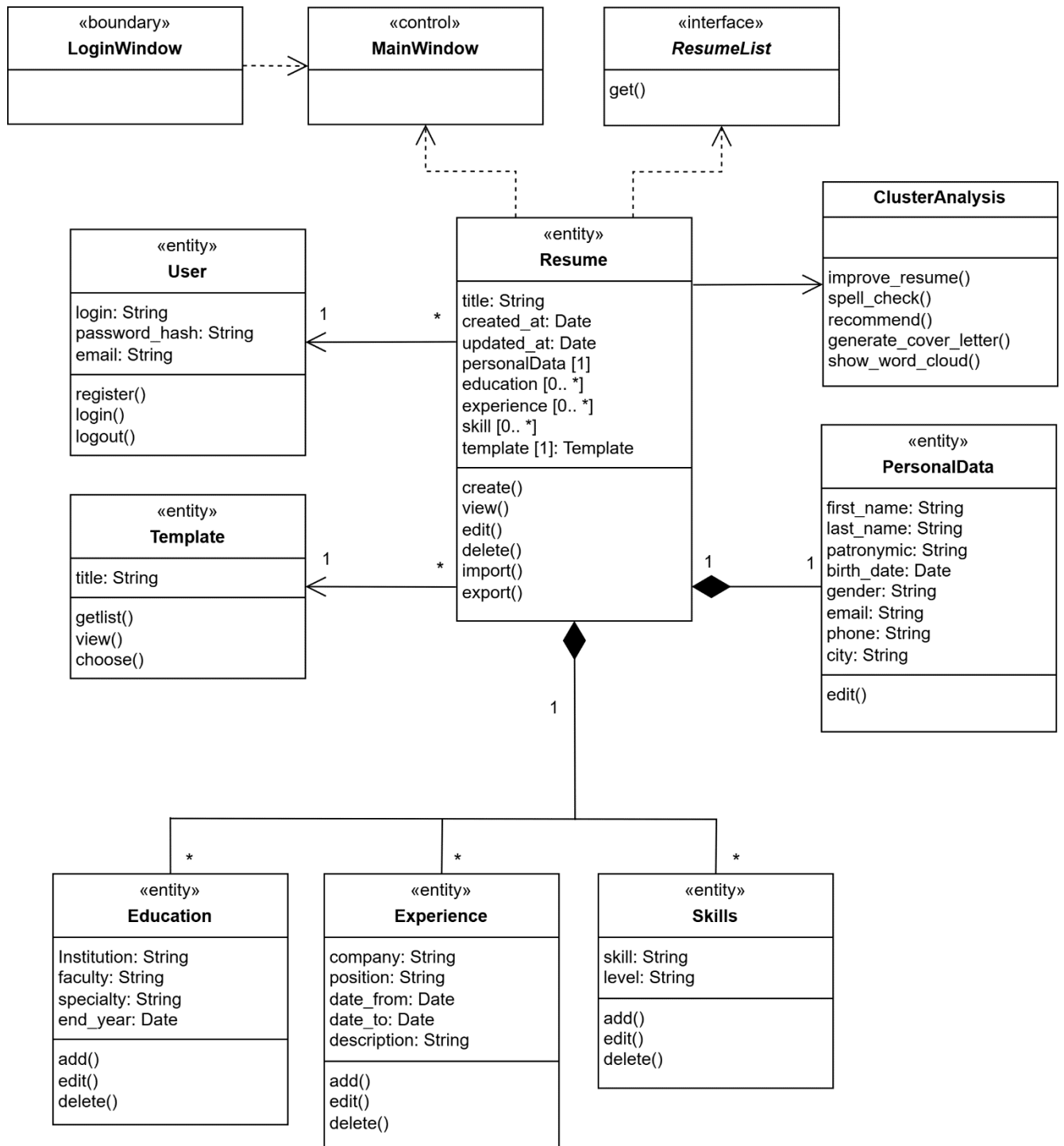


Рисунок 4 – Диаграмма классов

Модель User – это стандартная модель Django, которая предоставляет базовую функциональность для управления пользователями. Остальные модели определяются в приложении `models.py`.

В таблицах 2-6 приведено описание моделей.

Таблица 2 – Модель Resume

Название	Описание	Тип	Обязательность
title	Название резюме	CharField	Нет
created_at	Дата и время создания резюме	DateTimeField	Да
updated_at	Дата и время последнего обновления резюме	DateTimeField	Да

Таблица 3 – Модель PersonalData

Название	Описание	Тип	Обязательность
first_name	Имя	CharField	Да
last_name	Фамилия	CharField	Да
patronymic	Отчество	CharField	Нет
birth_date	Дата рождения	DateField	Да
gender	Пол	CharField	Да
email	Электронная почта	EmailField	Да
phone	Номер телефона	CharField	Да
city	Город проживания	CharField	Да

Таблица 4 – Модель Education

Название	Описание	Тип	Обязательность
institution	Название учебного заведения	CharField	Да
faculty	Факультет	CharField	Да
specialty	Специальность	CharField	Да
end_year	Год окончания	PositiveIntegerField	Да

Таблица 5 – Модель Experience

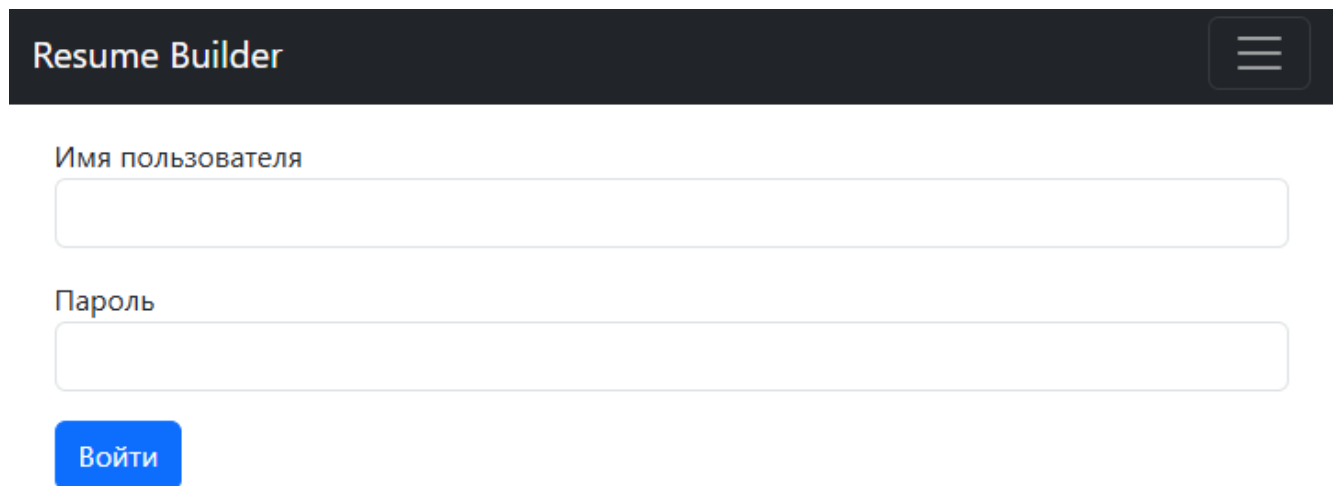
Название	Описание	Тип	Обязательность
company	Название компании	CharField	Да
position	Должность	CharField	Да
date_from	Дата начала работы	DateField	Да
date_to	Дата окончания работы	DateField	Нет
description	Описание обязанностей	TextField	Нет

Таблица 6 – Модель Skills

Название	Описание	Тип	Обязательность
skill	Навык	CharField	Да
level	Уровень владения	CharField	Да

2.2.5 Описание интерфейса

Форма аутентификации (рисунок 5) позволяет пользователю войти в систему, введя свои учетные данные (имя пользователя и пароль). После успешной аутентификации пользователь перенаправляется на главную страницу, где может управлять своими резюме.



Resume Builder

Имя пользователя

Пароль

Войти

Рисунок 5 – Форма аутентификации пользователя

Главная страница (рисунок 6) содержит список всех резюме пользователя. Для каждого резюме доступны кнопки **Просмотреть**, **Редактировать** и **Удалить**. Для нового пользователя отображается текстовое сообщение «Нет сохраненных резюме» (рисунок 7). Также на странице присутствует кнопка **Создать резюме**, по нажатию которой открывается форма создания резюме (рисунок 8).

Мои резюме

Резюме 3

[Просмотреть](#) [Редактировать](#) [Удалить](#)

Новое

[Просмотреть](#) [Редактировать](#) [Удалить](#)

Тестовое

[Просмотреть](#) [Редактировать](#) [Удалить](#)

Data Scientist

[Просмотреть](#) [Редактировать](#) [Удалить](#)

Тестовое


[Просмотреть](#) [Редактировать](#) [Удалить](#)

Пример резюме

[Просмотреть](#) [Редактировать](#) [Удалить](#)

[Создать резюме](#)

Рисунок 6 – Главная страница сайта

Resume Builder 

Мои резюме

Нет сохранённых резюме.

[Создать резюме](#)

Рисунок 7 – Главная страница сайта для нового пользователя

Создать резюме

Название резюме

Персональные данные

Имя


Фамилия

Отчество

Дата рождения

Пол

Электронная почта

Номер телефона

Город проживания

Образование

Добавить образование

Опыт работы

Добавить опыт работы

Навыки

Добавить навык

Сохранить

Рисунок 8 – Форма создания резюме

Форма создания резюме состоит из нескольких разделов:

- Персональные данные;
- Образование;
- Опыт работы;
- Навыки.

В разделе **Персональные данные** пользователь указывает свои ФИО, дату рождения, пол, контактные данные и город проживания.

Все разделы, кроме раздела **Персональные данные**, могут состоять из одного или нескольких одинаковых наборов полей. После завершения заполнения формы пользователь может сохранить резюме.

В разделе **Образование** (рисунок 9) пользователь указывает информацию об учебных заведениях, которые он окончил. Для каждого учебного заведения необходимо указать название, факультет, специальность и год окончания.

Образование

Учебное заведение

Факультет

Специальность

Год окончания

Удалить

Добавить образование

Рисунок 9 – Раздел Образование

В разделе **Опыт работы** (рисунок 10) пользователь указывает информацию о своих предыдущих местах работы. Для каждого места работы необходимо указать название компании, должность, период работы и описание рабочих обязанностей.

Опыт работы

Компания

Должность

Дата начала работы

Дата окончания работы

Описание

Удалить

Добавить опыт работы

Рисунок 10 – Раздел Опыт работы

В разделе Навыки (рисунок 11) пользователь указывает свои профессиональные навыки и уровень владения ими: базовый, средний или продвинутый.

Навыки

Навык

Уровень

Удалить

Добавить навык

Рисунок 11 – Раздел Навыки

Форма просмотра резюме (рисунок 12) позволяет пользователю увидеть готовое резюме в том виде, в котором оно будет экспортировано. На этой странице также доступны кнопки для редактирования и удаления резюме.

Тестовое резюме

Название резюме: Тестовое резюме

Персональные данные

Имя: Иван

Фамилия: Иванов

Отчество: Иванович

Дата рождения: Jan. 21, 2000

Пол: Мужской

Электронная почта: ivanov09@mail.ru

Номер телефона: 79167895456

Город проживания: Москва

Образование

Учебное заведение: Московский государственный университет

Год окончания: 2020

Факультет: Факультет вычислительной математики и кибернетики

Специальность: Прикладная математика и информатика

Опыт работы

Опыт работы не указан.

Навыки

Навык: Python

Уровень: Средний

Навык: SQL

Уровень: Средний

Редактировать

Удалить

Экспорт в PDF

Рисунок 12 – Форма просмотра резюме

Интерфейс конструктора резюме разработан с учетом удобства пользователя, предоставляя интуитивно понятные формы для аутентификации, создания, редактирования и просмотра резюме.

2.3 Разработка Системы

2.3.1 Подготовка среды разработки

Разработка сайта на Django подробно описана в [7]. Для начала необходимо подготовить среду разработки. Для изоляции зависимостей проекта от системных библиотек необходимо создать виртуальную среду:

```
virtualenv resume_env
```

После создания виртуального окружения его необходимо активировать:

```
resume_env\Scripts\activate
```

Далее необходимо установить Django:

```
pip install django
```

2.3.2 Создание и регистрация приложения

После установки Django можно создать новый проект:

```
django-admin startproject resume_builder
```

Чтобы работать с проектом, нужно перейти в его директорию:

```
cd resume_builder
```

Далее необходимо создать приложение resumes:

```
python manage.py startapp resumes
```

После создания приложения, необходимо зарегистрировать его в проекте, чтобы различные утилиты затрагивали его своим действием (например, при добавлении моделей в базу данных). Приложения регистрируются добавлением их названий в список `INSTALLED_APPS` (Листинг 1) в настройках проекта (`settings.py`).

Листинг 1 – Список приложений

```
INSTALLED_APPS = [  
    "django.contrib.admin",  
    "django.contrib.auth",  
    "django.contrib.contenttypes",  
    "django.contrib.sessions",  
    "django.contrib.messages",  
    "django.contrib.staticfiles",  
    "resumes",  
]
```

2.3.3 Создание суперпользователя

Для того, чтобы войти в административную панель, необходимо иметь учётную запись суперпользователя (администратора). Она создается с помощью команды:

```
python manage.py createsuperuser
```

Далее нужно указать имя суперпользователя, адрес электронной почты и надёжный пароль. В дальнейшем с помощью административной панели будут создаваться пользователи Системы.

2.3.4 Создание моделей и применение миграций

Спроектированные ранее модели добавляются в файл `models.py`, после чего необходимо создать миграции с помощью команды:

```
python manage.py makemigrations
```

и затем применить миграции:

```
python manage.py migrate
```

2.3.5 Создание представлений, форм и маршрутов

Для реализации функциональности системы были созданы представления в файле `views.py`. Далее для связывания URL-адресов с представлениями были настроены маршруты в файле `urls.py`. Для взаимодействия с пользователем и ввода данных были созданы формы, которые позволяют пользователю заполнять поля резюме. Формы определены в файле `forms.py` и связаны с моделями.

2.3.6 Создание шаблонов

Базовый шаблон `base.html` содержит общие элементы, такие как навигационная панель, подключение CSS и JavaScript, а также блоки для переопределения в дочерних шаблонах. На его основе были созданы шаблоны аутентификации, просмотра, создания, редактирования и удаления резюме, просмотра списка резюме, а также экспорта резюме в PDF.

2.3.7 Запуск сайта

После внесения всех изменений в проект необходимо запустить веб-сервер, используя следующую команду:

```
python manage.py runserver
```

Когда сервер запустится, можно перейти в веб-браузере на страницу аутентификации по адресу `http://127.0.0.1:8000/login/` и перейти к тестированию функциональности.

2.4 Тестирование Системы

2.4.1 Аутентификация пользователя

Предусловия

- С помощью административной панели Django в системе заведены пользователи.

Сценарий

1. Пользователь в адресной строке браузера вводит адрес домашней страницы сайта и нажимает на клавишу **Enter**. Отображается форма аутентификации пользователя.

2. Пользователь вводит корректные имя пользователя и пароль и нажимает на кнопку **Войти**.

Положительный результат

- Пользователь успешно аутентифицирован в системе.
- На странице отображается список резюме пользователя и кнопка **Создать резюме**. Для нового пользователя отображается текстовое сообщение «Нет сохраненных резюме» и кнопка **Создать резюме**.

2.4.2 Создание резюме

Предусловия

- Пользователь аутентифицирован в системе.

Сценарий

1. Пользователь нажимает на кнопку **Создать резюме**. Отображается форма создания резюме.

2. Пользователь указывает название резюме и заполняет поля раздела **Персональные данные**.

3. Пользователь нажимает на кнопку **Добавить образование**.
Отображаются поля раздела **Образование**.

4. Пользователь заполняет поля раздела **Образование**.

5. Пользователь повторяет шаги 3-4 необходимое количество раз.

6. Пользователь нажимает на кнопку **Добавить опыт работы**.
Отображаются поля раздела **Опыт работы**.

7. Пользователь заполняет поля раздела **Опыт работы**.

8. Пользователь повторяет шаги 6-7 необходимое количество раз.

9. Пользователь нажимает на кнопку **Добавить навык**. Отображаются поля
раздела **Навыки**.

10. Пользователь заполняет поля раздела **Навыки**.

11. Пользователь повторяет шаги 9-10 необходимое количество раз.

12. Пользователь нажимает на кнопку **Сохранить**.

Положительный результат

- Резюме успешно сохранено и отображается в списке резюме пользователя.

2.4.3 Просмотр резюме

Предусловия

- Пользователь аутентифицирован в системе.
- У пользователя имеются созданные ранее резюме.

Сценарий

1. Пользователь нажимает на кнопку **Просмотреть**, расположенную рядом с названием резюме в списке резюме. Отображается форма просмотра резюме.

2. После просмотра резюме пользователь нажимает на кнопку **Мои резюме**.
Отображается список резюме.

Положительный результат

- Пользователь может просмотреть резюме.

2.4.4 Редактирование резюме

Предусловия

- Пользователь аутентифицирован в системе.
- У пользователя имеются созданные ранее резюме.

Сценарий

1. Пользователь нажимает на кнопку **Редактировать** в форме просмотра резюме либо рядом с названием резюме в списке резюме. Отображается форма редактирования резюме.

2. Пользователь редактирует поля формы, добавляет и/или удаляет разделы резюме.

3. Пользователь нажимает на кнопку **Сохранить**.

Положительный результат

- Изменения успешно сохранены.

2.4.5 Удаление резюме

Предусловия

- Пользователь аутентифицирован в системе.
- У пользователя имеются созданные ранее резюме.

Сценарий

1. Пользователь нажимает на кнопку **Удалить** в форме просмотра резюме либо рядом с названием резюме в списке резюме. Отображается форма подтверждения удаления резюме.

2. Пользователь подтверждает удаление резюме. Отображается список резюме, в котором отсутствует удаленное резюме.

Положительный результат

- Резюме успешно удалено.

2.4.6 Экспорт резюме в PDF формат

Предусловия

- Пользователь аутентифицирован в системе.
- У пользователя имеются созданные ранее резюме.

Сценарий

1. Пользователь нажимает на кнопку **Просмотреть**, расположенную рядом с названием резюме в списке резюме. Отображается форма просмотра резюме.
2. Пользователь нажимает на кнопку **Экспорт в PDF**, расположенную в нижней части формы.
3. Резюме в формате PDF сохраняется в папке **Загрузки**.

Положительный результат

- Резюме успешно экспортировано в PDF формат.

Заключение

В рамках курсового проекта была разработана система конструктора резюме на базе фреймворка Django. В процессе работы были изучены основные компоненты Django, спроектирована логика работы Системы, реализованы и протестированы функции аутентификации, создания, редактирования и просмотра резюме, а также экспорта резюме в PDF.

В будущем можно расширить функциональность системы добавив возможность выбора шаблона резюме. Также предполагается реализовать кластерный анализ, подробно рассмотренный в [8], что позволит включить следующие функциональные возможности:

- Проверка правописания;
- Улучшение резюме;
- Генерация сопроводительного письма;
- Получение рекомендации.

Библиографический список

1. Бабич, А. Введение в UML / А. Бабич. – М.: ИНТУИТ, 2016. – 208 с.
2. Арлоу, Дж. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Дж. Арлоу, А. Нейштадт. – 2-е издание. – М.: Издательский дом «Вильямс», 2007. – 624 с.
3. Django [Электронный ресурс]. – Режим доступа: <https://www.djangoproject.com/> . – Заглавие с экрана. – (Дата обращения: 21.03.2025).
4. Постолиит, А.В. Python, Django и Bootstrap для начинающих / А.В. Постолиит. – СПб.: БХВ-Петербург, 2023. – 624 с.
5. Фаулер, М. UML. Основы / М. Фаулер. – 3-е издание. – СПб: Символ-Плюс, 2004. – 192 с.
6. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е издание. – СПб.: Питер, 2007. – 544 с.
7. Учебное пособие по созданию сайта локальной библиотеки с использованием Django [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn_web_development/Extensions/Server-side/Django/Tutorial_local_library_website. – Заглавие с экрана. – (Дата обращения: 21.03.2025).
8. Апельцин, Л. Data Science в действии / Л. Апельцин. – СПб.: Питер, 2023. – 736 с.

Приложение А

Код конструктора резюме

Модели (models.py)

```

from django.db import models
from django.contrib.auth.models import User

class Resume(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE,
verbose_name="Пользователь")
    name = models.CharField(max_length=100, verbose_name="Название
резюме", blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True,
verbose_name="Создано")
    updated_at = models.DateTimeField(auto_now=True,
verbose_name="Обновлено")

    def __str__(self):
        return f"{self.user.username}'s Resume #{self.id}"

class PersonalData(models.Model):
    resume = models.OneToOneField(Resume, on_delete=models.CASCADE,
related_name='personal_data')
    first_name = models.CharField(max_length=50, verbose_name="Имя")
    last_name = models.CharField(max_length=50,
verbose_name="Фамилия")
    patronymic = models.CharField(max_length=50,
verbose_name="Отчество", blank=True, null=True)
    birth_date = models.DateField(verbose_name="Дата рождения")
    GENDER_CHOICES = [
        ('M', 'Мужской'),
        ('F', 'Женский'),
    ]
    gender = models.CharField(max_length=1, choices=GENDER_CHOICES,
verbose_name="Пол")
    email = models.EmailField(verbose_name="Электронная почта")
    phone = models.CharField(max_length=16, verbose_name="Номер
телефона")
    city = models.CharField(max_length=100, verbose_name="Город
проживания")

class Education(models.Model):
    resume = models.ForeignKey(Resume, on_delete=models.CASCADE,
related_name='education_entries')
    institution = models.CharField(max_length=255,
verbose_name="Учебное заведение")

```

```

        faculty = models.CharField(max_length=255,
verbose_name="Факультет")
        specialty = models.CharField(max_length=255,
verbose_name="Специальность")
        end_year = models.PositiveIntegerField(verbose_name="Год
окончания")

class Experience(models.Model):
    resume = models.ForeignKey(Resume, on_delete=models.CASCADE,
related_name='experience_entries')
    company = models.CharField(max_length=255,
verbose_name="Компания")
    position = models.CharField(max_length=255,
verbose_name="Должность")
    date_from = models.DateField(verbose_name="Дата начала работы")
    date_to = models.DateField(verbose_name="Дата окончания работы",
blank=True, null=True)
    description = models.TextField(verbose_name="Описание",
blank=True, null=True)

class Skills(models.Model):
    resume = models.ForeignKey(Resume, on_delete=models.CASCADE,
related_name='skills_entries')
    skill = models.CharField(max_length=100, verbose_name="Навык")
    LEVEL_CHOICES = [
        ('B', 'Базовый'),
        ('I', 'Средний'),
        ('A', 'Продвинутый'),
    ]
    level = models.CharField(max_length=1, choices=LEVEL_CHOICES,
verbose_name="Уровень")

```

Маршруты (urls.py)

```
from django.urls import path, include
from . import views
from resumes.views import home
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('login/',
        auth_views.LoginView.as_view(template_name='resumes/login.html'),
        name='login'),
    path('logout/',
        auth_views.LogoutView.as_view(next_page='login'), name='logout'),
    path('my-resumes/', views.my_resumes, name='my_resumes'),
    path('resume/new/', views.resume_create, name='resume_create'),
    path('resume/<int:pk>/', views.resume_detail,
        name='resume_detail'),
    path('resume/<int:pk>/edit/', views.resume_edit,
        name='resume_edit'),
    path('resume/<int:pk>/delete/', views.resume_delete,
        name='resume_delete'),
    path('resume/<int:pk>/export-pdf/', views.resume_export_pdf,
        name='resume_export_pdf'),
]
```

Формы (forms.py)

```
from django import forms
from .models import Resume, PersonalData, Education, Experience,
Skills

class ResumeForm(forms.ModelForm):
    class Meta:
        model = Resume
        fields = ['title']

class PersonalDataForm(forms.ModelForm):
    class Meta:
        model = PersonalData
        fields = [
            'first_name', 'last_name', 'patronymic',
            'birth_date', 'gender', 'email', 'phone', 'city'
        ]

class EducationForm(forms.ModelForm):
    class Meta:
        model = Education
        fields = ['institution', 'faculty', 'specialty', 'end_year']

class ExperienceForm(forms.ModelForm):
    class Meta:
        model = Experience
        fields = ['company', 'position', 'date_from', 'date_to',
        'description']

class SkillsForm(forms.ModelForm):
    class Meta:
        model = Skills
        fields = ['skill', 'level']
```

Представления (views.py)

```

from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login
from django.contrib.auth.decorators import login_required
from .models import Resume, PersonalData, Education, Experience,
Skills
from .forms import ResumeForm, PersonalDataForm, EducationForm,
ExperienceForm, SkillsForm
from django.contrib.auth import views as auth_views
from django.contrib import messages
from django.http import HttpResponseRedirect
from django.template.loader import render_to_string
from weasyprint import HTML

def home(request):
    return render(request, 'base.html')

def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username,
password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Неверное имя пользователя или
пароль')
            return render(request, 'login.html')

    return render(request, 'login.html')

@login_required
def my_resumes(request):
    resumes = Resume.objects.filter(user=request.user)
    return render(request, 'resumes/my_resumes.html', {'resumes':
resumes})

@login_required
def resume_create(request):
    if request.method == 'POST':
        resume_form = ResumeForm(request.POST)
        personal_data_form = PersonalDataForm(request.POST)

        if resume_form.is_valid() and personal_data_form.is_valid():
            # Сохраняем резюме

```

```

resume = resume_form.save(commit=False)
resume.user = request.user
resume.save()

# Сохраняем персональные данные
personal_data = personal_data_form.save(commit=False)
personal_data.resume = resume
personal_data.save()

# Обработка полей образования
education_count = 0
while f"institution_{education_count}" in request.POST:
    institution =
request.POST.get(f"institution_{education_count}")
    faculty =
request.POST.get(f"faculty_{education_count}")
    specialty =
request.POST.get(f"specialty_{education_count}")
    end_year =
request.POST.get(f"end_year_{education_count}")

    if institution or faculty or specialty or end_year:
        Education.objects.create(
            resume=resume,
            institution=institution,
            faculty=faculty,
            specialty=specialty,
            end_year=end_year
        )
    education_count += 1

# Обработка полей опыта работы
experience_count = 0
while f"company_{experience_count}" in request.POST:
    company =
request.POST.get(f"company_{experience_count}")
    position =
request.POST.get(f"position_{experience_count}")
    date_from =
request.POST.get(f"date_from_{experience_count}")
    date_to =
request.POST.get(f"date_to_{experience_count}")
    description =
request.POST.get(f"description_{experience_count}")

    if company or position or date_from or date_to or
description:
        Experience.objects.create(
            resume=resume,
            company=company,
            position=position,
            date_from=date_from,
            date_to=date_to,

```

```

        description=description
    )
    experience_count += 1

    # Обработка полей НАВЫКОВ
    skills_count = 0
    while f"skill_{skills_count}" in request.POST:
        skill = request.POST.get(f"skill_{skills_count}")
        level = request.POST.get(f"level_{skills_count}")

        if skill and level:
            Skills.objects.create(
                resume=resume,
                skill=skill,
                level=level
            )
            skills_count += 1

    return redirect('my_resumes')
else:
    resume_form = ResumeForm()
    personal_data_form = PersonalDataForm()

    return render(request, 'resumes/create_resume.html', {
        'resume_form': resume_form,
        'personal_data_form': personal_data_form
    })

@login_required
def resume_detail(request, pk):
    resume = get_object_or_404(Resume, pk=pk, user=request.user)
    personal_data = resume.personal_data
    education_entries = resume.education_entries.all()
    experience_entries = resume.experience_entries.all()
    skills_entries = resume.skills_entries.all()

    return render(request, 'resumes/resume_detail.html', {
        'resume': resume,
        'personal_data': personal_data,
        'education_entries': education_entries,
        'experience_entries': experience_entries,
        'skills_entries': skills_entries,
    })

@login_required
def resume_edit(request, pk):
    resume = get_object_or_404(Resume, pk=pk, user=request.user)
    personal_data = resume.personal_data
    education_entries = resume.education_entries.all()
    experience_entries = resume.experience_entries.all()
    skills_entries = resume.skills_entries.all()

    if request.method == 'POST':

```

```

resume_form = ResumeForm(request.POST, instance=resume)
personal_data_form = PersonalDataForm(request.POST,
instance=personal_data)

if resume_form.is_valid() and personal_data_form.is_valid():
    resume_form.save()
    personal_data_form.save()

    # Обработка существующих записей об образовании
    for education in education_entries:
        delete_key = f"delete_education_{education.pk}"
        if delete_key in request.POST:
            education.delete()
        else:
            education.institution =
request.POST.get(f"institution_{education.pk}")
            education.faculty =
request.POST.get(f"faculty_{education.pk}")
            education.specialty =
request.POST.get(f"specialty_{education.pk}")
            education.end_year =
request.POST.get(f"end_year_{education.pk}")
            education.save()

    # Обработка новых записей об образовании
    education_count = education_entries.count()
    while True:
        institution =
request.POST.get(f"institution_{education_count}")
        faculty =
request.POST.get(f"faculty_{education_count}")
        specialty =
request.POST.get(f"specialty_{education_count}")
        end_year =
request.POST.get(f"end_year_{education_count}")

        if not (institution or faculty or specialty or
end_year):
            break

        Education.objects.create(
            resume=resume,
            institution=institution,
            faculty=faculty,
            specialty=specialty,
            end_year=end_year
        )
        education_count += 1

    # Обработка существующих записей об опыте работы
    for experience in experience_entries:
        delete_key = f"delete_experience_{experience.pk}"
        if delete_key in request.POST:

```

```

        experience.delete()
    else:
        experience.company =
request.POST.get(f"company_{experience.pk}")
        experience.position =
request.POST.get(f"position_{experience.pk}")
        experience.date_from =
request.POST.get(f"date_from_{experience.pk}")
        experience.date_to =
request.POST.get(f"date_to_{experience.pk}")
        experience.description =
request.POST.get(f"description_{experience.pk}")
        experience.save()

    # Обработка новых записей об опыте работы
    experience_count = experience_entries.count()
    while True:
        company =
request.POST.get(f"company_{experience_count}")
        position =
request.POST.get(f"position_{experience_count}")
        date_from =
request.POST.get(f"date_from_{experience_count}")
        date_to =
request.POST.get(f"date_to_{experience_count}")
        description =
request.POST.get(f"description_{experience_count}")

        if not (company or position or date_from or date_to
or description):
            break

    Experience.objects.create(
        resume=resume,
        company=company,
        position=position,
        date_from=date_from,
        date_to=date_to,
        description=description
    )
    experience_count += 1

    # Обработка существующих записей о навыках
    for skill in skills_entries:
        delete_key = f"delete_skill_{skill.pk}"
        if delete_key in request.POST:
            skill.delete()
        else:
            skill.skill =
request.POST.get(f"skill_{skill.pk}")
            skill.level =
request.POST.get(f"level_{skill.pk}")
            skill.save()

```

```

# Обработка новых записей о навыках
skills_count = skills_entries.count()
while True:
    skill = request.POST.get(f"skill_{skills_count}")
    level = request.POST.get(f"level_{skills_count}")

    if not (skill and level):
        break

    Skills.objects.create(
        resume=resume,
        skill=skill,
        level=level
    )
    skills_count += 1

    return redirect('my_resumes')
else:
    resume_form = ResumeForm(instance=resume)
    personal_data_form =
PersonalDataForm(instance=personal_data)

    return render(request, 'resumes/edit_resume.html', {
        'resume_form': resume_form,
        'personal_data_form': personal_data_form,
        'education_entries': education_entries,
        'experience_entries': experience_entries,
        'skills_entries': skills_entries,
        'resume': resume,
    })

@login_required
def resume_delete(request, pk):
    resume = get_object_or_404(Resume, pk=pk, user=request.user)
    if request.method == 'POST':
        resume.delete()
        return redirect('my_resumes')
    return render(request, 'resumes/delete_resume.html', {'resume':
resume})

@login_required
def resume_export_pdf(request, pk):
    # Получаем резюме и связанные данные
    resume = get_object_or_404(Resume, pk=pk, user=request.user)
    personal_data = resume.personal_data
    education_entries = resume.education_entries.all()
    experience_entries = resume.experience_entries.all()
    skills_entries = resume.skills_entries.all()

    # Рендерим HTML-шаблон с данными резюме
    html_string = render_to_string('resumes/resume_pdf.html', {
        'resume': resume,

```

```

        'personal_data': personal_data,
        'education_entries': education_entries,
        'experience_entries': experience_entries,
        'skills_entries': skills_entries,
    })

    # Конвертируем HTML в PDF с помощью WeasyPrint
    html = HTML(string=html_string,
base_url=request.build_absolute_uri('/'))
    pdf = html.write_pdf()

    # Возвращаем PDF как HTTP-ответ
    response = HttpResponse(pdf, content_type='application/pdf')
    response['Content-Disposition'] = f'attachment;
filename="resume_{resume.pk}.pdf"'
    return response

```