**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.02 - Информационные системы и технологии | |
| **Профиль** | Информационные системы и технологии в бизнесе | |
| **Факультет** | КТИ | |
| **Кафедра** | АПУ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Шестопалов М.Ю. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

Тема: Разработка приложения для рекомендаций по выбору исполнителей в IT-проекте

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Какраев М.С. |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | Писарев А.С. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты |  |  |  | Шевнева И. В. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  |  |  |  | Ряскова Е.Б. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | к.т.н., доцент |  |  | Белаш О.Ю. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой АПУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шестопалов М.Ю. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Какраев М.С. | | | |  | Группа | 1371 | |
| Тема работы: Разработка приложения для рекомендаций по выбору исполнителей в IT-проекте | | | | | | | | |
| Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им. В.И.Ульянова (Ленина) | | | | | | | | |
| Исходные данные (технические требования):  В рамках выполнения ВКР необходимо спроектировать и разработать веб-приложение для автоматизированного подбора кандидатов на основе анализа текста вакансий и резюме. ВКР включает описание архитектуры клиент-серверного приложения, реализацию взаимодействия между фронтендом и бэкендом, а также разработку алгоритма семантического сравнения текстов с использованием моделей машинного обучения. Особое внимание уделяется обеспечению удобного пользовательского интерфейса для добавления вакансий и резюме, а также отображению наиболее релевантных кандидатов. | | | | | | | | |
| Содержание ВКР:  Описание целей и задач работы, анализ и сравнение существующих систем, поиск существующих алгоритмов формирования рекомендаций, разработка архитектуры приложения, реализация алгоритмов формирования рекомендаций, реализация интерфейса и логики веб-приложения тестирование веб-приложения | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал, демо-ролик с представлением работы проекта  Дополнительный раздел: экономическое обоснование | | | | | | | | |
|  | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «1» апреля 2025 г. | | | «18» июня 2025 г. | | | | | |
|  | | |  | | | | | |
| Студент | |  | | Какраев М.С. | | | |
| Руководитель к.т.н., доцент | |  | | Писарев А.С. | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой АПУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шестопалов М.Ю. |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Какраев М.С. |  | Группа | 1371 |
| Тема работы: Разработка приложения для рекомендаций по выбору исполнителей в IT-проекте | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Поиск литературы по теме работы, постановка целей и задач | 01.04 – 08.04 |
| 2 | Анализ и сравнение существующих систем | 09.04 – 15.04 |
| 3 | Поиск существующих алгоритмов формирования рекомендаций и их реализация | 16.04 – 20.04 |
| 4 | Разработка архитектуры приложения | 21.04 – 25.04 |
| 5 | Реализация интерфейса и логики веб-приложения | 26.04 – 09.05 |
| 6 | Тестирование веб-приложения | 10.05 – 19.05 |
| 7 | Оформление пояснительной записки | 20.05 – 29.05 |
| 8 | Подготовка к защите | 30.05 – 17.06 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Какраев М.С. |
| Руководитель к.т.н., доцент |  | Писарев А.С. |

**РЕФЕРАТ**

Пояснительная записка содержит: 59 стр., 16 рис., 6 табл., 16 источников, 2 прил.

Ключевые слова: ВЕБ-ПРИЛОЖЕНИЕ, РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, СЕМАНТИЧЕСКОЕ СРАВНЕНИЕ, EMBEDDING, MACHINE LEARNING, DJANGO.

Цель работы: Разработка веб-приложения для автоматизированного подбора кандидатов на основе анализа текста вакансий и резюме с применением семантического сравнения.

В работе рассматривается архитектура клиент-серверного приложения с фронтендом на HTML/JavaScript и бэкендом на Django. Особое внимание уделено алгоритму ранжирования кандидатов, основанному на сравнении эмбеддингов вакансий и резюме с использованием модели SentenceTransformer на базе LaBSE-ru-turbo и функции потерь MultipleNegativesRankingLoss. Проведен анализ ключевых компетенций для выбранной предметной области — роли Project Manager.

В практической части выполнена реализация прототипа веб-приложения с возможностями добавления вакансий и резюме, а также выдачи топ-5 наиболее релевантных кандидатов. Оценена эффективность алгоритма на тестовых данных, подтверждена его работоспособность и устойчивость.

Рассчитана себестоимость разработки и приведен анализ структуры затрат. Разработанное приложение может быть использовано для автоматизации процесса подбора персонала и дальнейшего развития систем рекрутинга с применением методов машинного обучения.

**ABSTRACT**

The explanatory note contains: 59 pages, 16 figures, 6 tables, 16 references, 2 appendices.

Keywords: WEB APPLICATION, RECOMMENDATION SYSTEM, SEMANTIC COMPARISON, EMBEDDING, MACHINE LEARNING, DJANGO.

Objective: Development of a web application for automated candidate selection based on the analysis of vacancy and resume texts using semantic comparison.

The work covers the architecture of a client-server application with a frontend implemented in HTML/JavaScript and a backend in Django. Special attention is given to the candidate ranking algorithm based on comparing embeddings of vacancies and resumes using the SentenceTransformer model built on LaBSE-ru-turbo and the MultipleNegativesRankingLoss loss function. A key competencies analysis was conducted for the selected domain — the role of Project Manager.

The practical part includes the implementation of a prototype web application with features for adding vacancies and resumes and providing the top-5 most relevant candidates. The algorithm’s effectiveness was evaluated on test data, confirming its functionality and robustness.

The development cost was calculated, and the cost structure analyzed. The developed application can be used to automate the recruitment process and further advance recruiting systems employing machine learning methods.

**СОДЕРЖАНИЕ**

[*Определения, обозначения и сокращения 8*](#_Toc200624222)

[*ВВЕДЕНИЕ 12*](#_Toc200624223)

[*1. Технологии и методы построения рекомендательных систем 14*](#_Toc200624224)

[*1.1. Контентная фильтрация 14*](#_Toc200624225)

[*1.2. Коллаборативная фильтрация 14*](#_Toc200624226)

[*1.3. Гибридные методы 15*](#_Toc200624227)

[*1.4. Другие современные подходы: машинное и глубинное обучение 15*](#_Toc200624228)

[*2. Прогноз изменения востребованности IT-специалистов в условиях распространения искусственного интеллекта 16*](#_Toc200624229)

[*2.1. Замена человека искусственным интеллектом 16*](#_Toc200624230)

[*2.2. Появление новых профессий в сфере ИИ 18*](#_Toc200624231)

[*2.3. Актуальная ситуация на российском рынке IT-вакансий 18*](#_Toc200624232)

[*3. Проектирование рекомендательной системы на основе семантического сопоставления текстов 20*](#_Toc200624233)

[*3.1. Выбор профессиональной роли для построения рекомендательной модели 20*](#_Toc200624234)

[*3.2. Формализация задачи подбора по смысловому сходству 20*](#_Toc200624235)

[*3.3. Выбор модели и обоснование архитектурного решения 20*](#_Toc200624236)

[*3.4. Алгоритм обучения и функция потерь 21*](#_Toc200624237)

[*3.5. Подготовка обучающих данных 21*](#_Toc200624238)

[*4. Архитектура и реализация веб-приложения 22*](#_Toc200624239)

[*4.1. Общая архитектура и логика взаимодействия компонентов 22*](#_Toc200624240)

[*4.2. Серверная часть: Django и структура API 25*](#_Toc200624241)

[*4.3. Клиентская часть: HTML/JS и пользовательские сценарии 26*](#_Toc200624242)

[*4.4. Модель данных и структура базы (SQLite) 26*](#_Toc200624243)

[*4.5. Интеграция ML-модуля с веб-приложением 27*](#_Toc200624244)

[*4.6. Разработанные прототипы интерфейса 28*](#_Toc200624245)

[*4.6.1. Главная страница (домашняя) 28*](#_Toc200624246)

[*4.6.2. Страница авторизации (логин) 28*](#_Toc200624247)

[*4.6.3. Административная страница 29*](#_Toc200624248)

[*5. Функциональные возможности системы 31*](#_Toc200624249)

[*5.1. Авторизация и разграничение доступа 31*](#_Toc200624250)

[*5.2. Интерфейс отправки резюме (для неавторизованных пользователей) 32*](#_Toc200624251)

[*5.3. Интерфейс администратора: управление вакансиями и резюме 32*](#_Toc200624252)

[*5.4. Встроенная административная панель Django 35*](#_Toc200624253)

[*6. Реализация и тестирование 38*](#_Toc200624254)

[*6.1. Этапы разработки и технологии реализации 38*](#_Toc200624255)

[*6.2. Тестирования алгоритма подбора кандидатов 38*](#_Toc200624256)

[*6.2.1. Методика тестирования 39*](#_Toc200624257)

[*6.2.2. Описание тестовой вакансии и групп кандидатов 39*](#_Toc200624258)

[*6.2.3. Выбранные метрики 40*](#_Toc200624259)

[*6.2.4. Оценка результатов тестирования 41*](#_Toc200624260)

[*7. Экономическое обоснование 44*](#_Toc200624261)

[*7.1. Концепция 44*](#_Toc200624262)

[*7.2. Расчет затрат на оплату труда 44*](#_Toc200624263)

[*7.3. Расчет накладных расходов 47*](#_Toc200624264)

[*7.4. Расходы на материалы 47*](#_Toc200624265)

[*7.5. Издержки на амортизацию ПК и оргтехники 48*](#_Toc200624266)

[*7.6. Расходы на услуги сторонних организаций 48*](#_Toc200624267)

[*7.7. Полная себестоимость работы 49*](#_Toc200624268)

[*7.8. Вывод 49*](#_Toc200624269)

[*Заключение 51*](#_Toc200624270)

[*СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 53*](#_Toc200624271)

[*Приложение A. Python: описание Django моделей 57*](#_Toc200624272)

[*Приложение Б. Python: Модуль, отвечающий за ML 59*](#_Toc200624273)

# Определения, обозначения и сокращения

В настоящей работе применяют следующие термины с соответствующими определениями.

|  |  |
| --- | --- |
| API (Application Programming Interface) | — набор определённых правил и спецификаций, позволяющий программному обеспечению взаимодействовать друг с другом |
| ALS (Alternating Least Squares) | — aлгоритм матричной факторизации, часто применяемый в коллаборативной фильтрации для рекомендаций на основе предпочтений пользователей |
| Bi-encoder | — aрхитектура нейросетевой модели, в которой два входа (например, резюме и вакансия) обрабатываются независимо и сопоставляются по косинусному сходству в эмбеддинговом пространстве |
| CI/CD (Continuous Integration / Continuous Deployment) | — методологии автоматизированной сборки, тестирования и развёртывания программного обеспечения, обеспечивающие быструю доставку изменений в продуктивную среду |
| Deep Learning (Глубинное обучение) | — раздел машинного обучения, использующий многослойные нейронные сети для выявления сложных закономерностей в данных |
| DNN (Deep Neural Network, Глубокая нейронная сеть) | искусственная нейронная сеть с множеством скрытых слоёв, способная обучаться сложным паттернам в данных. Используется в компьютерном зрении, NLP и других областях ИИ |
| Embedding (Эмбеддинг) | — представление слов, предложений или документов в виде векторов фиксированной размерности, отражающих их смысловое содержание |
| GitHub | — платформа для хостинга IT-проектов и совместной разработки на основе Git. |
| GPT (Generative Pre-trained Transformer) | — серия моделей ИИ для генерации текста, основанных на архитектуре Transformer. Обучены на огромных объёмах данных и способны выполнять задачи NLP (перевод, ответы на вопросы и т. д.) |
| hh.ru (HeadHunter) | — крупнейшая в России платформа для поиска работы и подбора персонала. |
| HR (Human Resources, Отдел кадров) | — подразделение компании, отвечающее за подбор персонала, адаптацию сотрудников, кадровую политику и управление трудовыми ресурсами |
| Machine Learning (Машинное обучение, ML) | — методы анализа данных, позволяющие системам самостоятельно обучаться на основе примеров и делать прогнозы без явного программирования |
| MultipleNegativesRankingLoss | — функция потерь, используемая в обучении моделей сопоставления текстов, оптимизирующая расстояние между положительными и отрицательными парами в батче |
| MTEB-бенчмарк (Massive Text Embedding Benchmark) | — тестовый набор данных для оценки качества моделей, создающих векторные представления текста (эмбеддинги). Включает разнообразные задачи: классификацию, кластеризацию, поиск и т. д. |
| NLP (Natural Language Processing, Обработка естественного языка) | — раздел ИИ, изучающий взаимодействие компьютеров и человеческого языка. Включает машинный перевод, генерацию текста, анализ тональности и др. |
| ORM Django (Object-Relational Mapping) | — технология в веб-фреймворке Django, позволяющая работать с базой данных через Python-объекты вместо SQL-запросов. |
| Project Manager (PM) | — cпециалист, управляющий проектами, координирующий команду и ресурсы, отслеживающий сроки, бюджет и цели |
| Recommender System (Рекомендательная система) | — информационная система, предлагающая пользователю персонализированные рекомендации на основе анализа данных о его предпочтениях и поведении. |
| REST API | — архитектурный стиль взаимодействия компонентов распределённой системы по протоколу HTTP с использованием стандартных методов (GET, POST и др.) |
| SVD (Singular Value Decomposition) | — метод матричной факторизации, используемый в коллаборативной фильтрации для выявления скрытых факторов предпочтений |
| SentenceTransformer | — модель машинного обучения, преобразующая предложения в эмбеддинги, пригодные для оценки семантического сходства |
| TF-IDF | — статистическая мера, оценивающая важность слова в документе относительно корпуса текстов, часто используется в информационном поиске |
| TDD (Test-Driven Development) | — практика разработки, при которой сначала пишутся тесты, а затем код, обеспечивающий их прохождение. |
| Word2Vec | — алгоритм обучения словесных эмбеддингов, представляющий слова в виде векторов, близких по смыслу |
| Батч (Batch) | — группа данных, обрабатываемая одновременно в машинном обучении или при работе с базами данных. Уменьшает нагрузку на систему за счёт пакетной обработки |

# ВВЕДЕНИЕ

***Обоснование актуальности темы***

В современных условиях цифровизации и развития онлайн-платформ для подбора персонала (таких как hh.ru, rabota.ru) наблюдается активное внедрение платных инструментов для продвижения вакансий. Это создает дополнительную конкуренцию среди работодателей, усложняя процесс поиска и привлечения квалифицированных кандидатов, особенно для компаний с ограниченным бюджетом на рекрутинг.

Кроме того, некоторые организации размещают вакансии на собственных корпоративных сайтах, предлагая соискателям отправлять резюме напрямую. Однако отсутствие автоматизированных механизмов обработки этих данных снижает эффективность подбора персонала, увеличивая временные затраты рекрутеров на анализ и сопоставление резюме с актуальными вакансиями.

Разработка веб-приложения, которое позволит автоматизировать процесс подбора кандидатов с использованием методов рекомендательных систем, может значительно повысить эффективность HR-отделов. Такой инструмент обеспечит работодателей релевантными рекомендациями специалистов, сокращая затраты на поиск персонала и улучшая процесс найма.

***Цели и задачи ВКР***

Целью является разработка веб-приложения на основе рекомендательной системы, которое позволит автоматизировать подбор наиболее подходящих исполнителей для IT-проектов, учитывая их профессиональные навыки, опыт и требования заказчика.

Задачи:

1. Провести анализ существующих рекомендательных систем и технологий их построения.
2. Определить ключевые характеристики исполнителей IT-проектов и параметры рекомендательной системы.
3. Выбрать математические модели и алгоритмы для формирования рекомендаций.
4. Разработать архитектуру веб-приложения, включая базу данных и основные функциональные модули.
5. Реализовать алгоритмы формирования рекомендаций с учетом характеристик исполнителей и требований проектов.
6. Разработать пользовательский интерфейс и логику веб-приложения.
7. Провести тестирование системы, оценить точность рекомендаций и эффективность работы приложения.
8. Подготовить документацию, проанализировать результаты работы системы и определить направления для дальнейшего развития.

***Литературный обзор***

* *“Fairness of Recommender Systems in the Recruitment Domain: An Analysis from Technical and Legal Perspectives”*

*Тематика:* В этом источнике рассматриваются вопросы справедливости и потенциальных предвзятостей в рекомендательных системах, применяемых в области рекрутинга. Авторы анализируют технические методы устранения дискриминации, а также юридические аспекты использования подобных систем.

* *“Applied Recommender Systems with Python: Build Recommender Systems with Deep Learning, NLP, and Graph-Based Techniques”*

*Тематика:* Практическое руководство по разработке рекомендательных систем с применением Python, глубокого обучения (Deep Learning), обработки естественного языка (NLP) и графовых методов.

* *“Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”*

*Тематика:* Один из наиболее цитируемых обзоров по рекомендательным системам, в котором рассматриваются классические и современные подходы, такие как коллаборативная фильтрация, контентные и гибридные методы.

# Технологии и методы построения рекомендательных систем

Разработка рекомендательных систем для подбора персонала в IT-проектах опирается на различные методы, среди них можно выделить: искусственный интеллекта (AI), машинное обучение (ML), глубинное обучение (Deep Learning), а также классические подходы. Эти технологии позволяют анализировать большие объемы данных о кандидатах и вакансиях, выявляя наиболее релевантные совпадения. Среди множества подходов, применяемых в рекомендательных системах, можно выделить три ключевых метода, наиболее подходящих для задач подбора IT-специалистов: контентная фильтрация, коллаборативная фильтрация и гибридные модели.

### **Контентная фильтрация**

Метод основан на анализе характеристик кандидата (навыков, опыта, образования) и их сравнении с требованиями вакансии. В книге “Applied Recommender Systems with Python” [2] рассматриваются алгоритмы обработки естественного языка (NLP), которые позволяют анализировать текст вакансий и резюме, выявляя ключевые компетенции и соответствия.

Примеры технологий:

* TF-IDF и Word2Vec – для обработки текстовых данных резюме и вакансий;
* Классификационные модели (например, Random Forest, XGBoost) – для оценки соответствия кандидата вакансии.

### **Коллаборативная фильтрация**

Метод использует данные о взаимодействии пользователей с системой. Например, анализирует, какие вакансии просматривают кандидаты, какие отклики оставляют работодатели.

Основные алгоритмы:

* Методы на основе матричного разложения (SVD, ALS).
* Графовые модели – строят связи между пользователями и объектами (вакансиями).

### **Гибридные методы**

Современные HR-системы часто комбинируют контентную и коллаборативную фильтрацию для повышения точности рекомендаций. В работе “*Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions” [3]* отмечается, что гибридные модели позволяют уменьшить недостатки каждого метода и повысить качество рекомендаций.

### **Другие современные подходы: машинное и глубинное обучение**

Помимо традиционных методов, активно применяются алгоритмы машинного обучения (ML) и глубинного обучения (Deep Learning), которые способны находить сложные закономерности в данных. Глубокие нейросети (DNN) позволяют учитывать не только текстовые характеристики резюме, но и скрытые зависимости между навыками, опытом кандидатов и требованиями вакансий.

Применение гибридных систем включает:

* Глубокие нейросети (Deep Learning) – позволяют учитывать сложные зависимости между характеристиками кандидатов и требованиями вакансий.
* Графовые модели (Graph-Based Recommender Systems) – используются для построения связей между кандидатами, проектами и работодателями.

# Прогноз изменения востребованности IT-специалистов в условиях распространения искусственного интеллекта

### **Замена человека искусственным интеллектом**

В последние годы в профессиональном и научном сообществе активно обсуждается вопрос возможной замены программистов искусственным интеллектом (ИИ). Это связано с быстрым развитием генеративных моделей, таких как GPT, и появлением интеллектуальных помощников, способных выполнять задачи, ранее доступные только человеку.

Сторонники технологического прорыва считают, что в ближайшем будущем ИИ сможет взять на себя практически весь процесс написания кода. Так, генеральный директор технологической компании Anthropic Дарио Амодеи в рамках выступления на Совете по международным отношениям (Council on Foreign Relations) заявил: *«Я думаю, через три-шесть месяцев ИИ будет писать 90% кода. А через 12 месяцев — практически весь код»*.

Однако ряд исследований и практических экспериментов демонстрируют, что пока такие заявления являются преждевременными. Например, проект Devin, разработанный Cognition Labs, позиционируется как ИИ-разработчик, способный осваивать незнакомые технологии, работать с чужими репозиториями, исправлять ошибки и даже тренировать собственные модели. Несмотря на это, согласно техническому отчету SWE-bench от Cognition, Devin смог успешно решить только 13,86% задач из open-source проектов на GitHub без участия человека. Для сравнения, квалифицированный разработчик способен справиться со 100% подобных задач. Таким образом, ИИ пока пригоден лишь для решения ограниченного класса задач под контролем специалиста. [5]

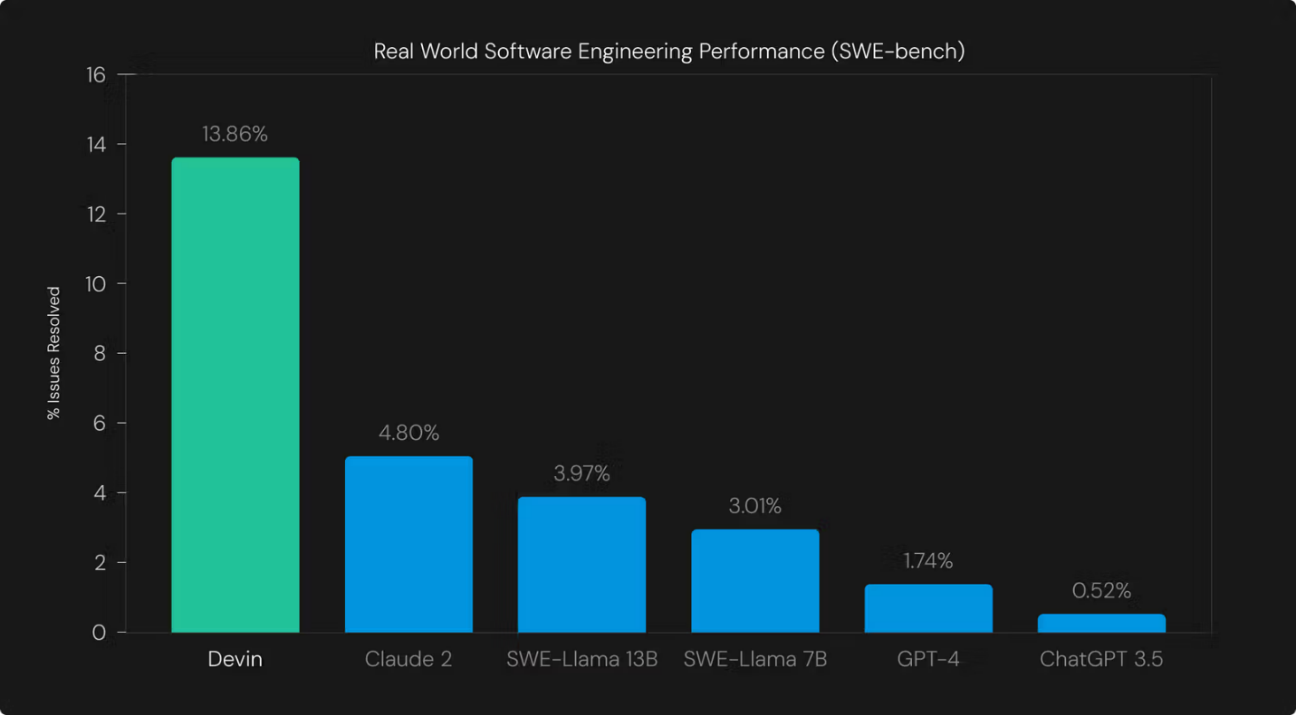


Рисунок 1- Сравнение различных ИИ по числу выполненных задач [5]

В то же время, даже при ограниченной автономности ИИ показывает значительный рост эффективности. Согласно отчету PwC’s 2024 AI Jobs Barometer, применение искусственного интеллекта приводит к увеличению производительности труда в 4,8 раза по сравнению с другими секторами. Кроме того, с 2016 года наблюдается рост числа вакансий, связанных с ИИ, в 3,5 раза быстрее, чем общий рост числа вакансий [4].

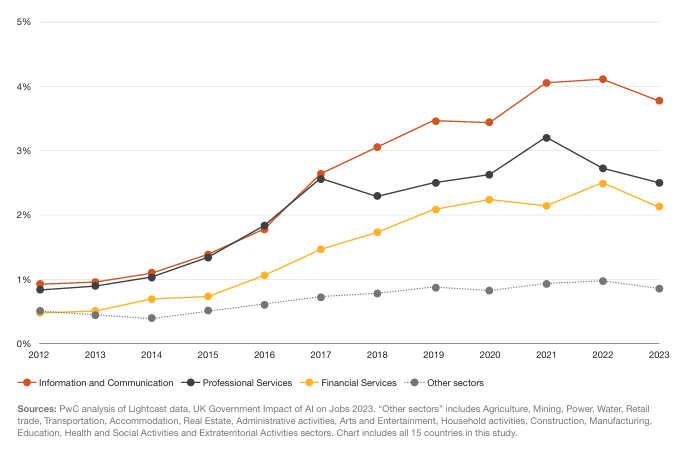


Рисунок 2- Рост производительности труда в различных отраслях [5]

### **Появление новых профессий в сфере ИИ**

На фоне растущего влияния искусственного интеллекта происходит формирование новых профессий, ориентированных на его разработку, настройку и сопровождение. По данным совместного исследования **hh.ru** и **МТС Линк**, число вакансий, связанных с ИИ, в России удвоилось за последние два года. В 2024 году на hh.ru было опубликовано более **3,7 тыс. вакансий**, в то время как в 2021 году — **1,7 тыс.** Анализ проводился на основе данных за период с января 2021 по август 2024 года. [6]

К числу новых профессий, связанных с ИИ, можно отнести:

* промпт-инженер;
* редактор и тренер текстов для нейросетей;
* специалист по разметке и аннотированию данных;
* эксперт по ИИ и координатор ИИ-проектов;
* переводчик для нейросетей.

Это указывает на трансформацию рынка труда, в которой появляются компетенции, ранее отсутствующие в классических IT-направлениях.

### **Актуальная ситуация на российском рынке IT-вакансий**

Согласно аналитике портала **hh.ru** за 2024 год, наибольшее количество вакансий в области информационных технологий относится к направлениям управления и сопровождения разработки. Так, вакансий на позицию **Project Manager** публиковалось в среднем около **4000** ежемесячно. Далее следуют должности **системного администратора** (около **3750**) и **сервисного инженера** (приблизительно **2000**).

По состоянию на апрель 2025 года:

* по запросу *«Project Manager»* на hh.ru размещено **около 4100** вакансий;
* по запросу *«Системный администратор»* — **около 2200**;
* по запросу *«Сервисный инженер»* — **примерно 1400**.

Данные тенденции отражают динамику изменения спроса на конкретные IT-специальности и могут быть использованы при формировании рекомендаций в рамках разрабатываемой системы подбора персонала.

# Проектирование рекомендательной системы на основе семантического сопоставления текстов

### **Выбор профессиональной роли для построения рекомендательной модели**

В данной работе в качестве целевой роли выбран **Project Manager** — специалист, отвечающий за координацию и управление IT-проектами.

Выбор данной профессии обусловлен несколькими факторами:

* высокая востребованность на рынке труда (по данным hh.ru, 2024–2025 годы показывают устойчивый рост числа вакансий в категории «Управление разработкой»);
* наличие устойчиво сформированных требований в вакансиях;
* баланс между управленческими и техническими навыками, что делает профиль специалиста подходящим для формализации;

### **Формализация задачи подбора по смысловому сходству**

Система рассматривает задачу подбора кандидатов как задачу семантического сопоставления двух текстов: описания вакансии и резюме. Задача решается путём вычисления степени смысловой близости между ними. Такой подход позволяет не ограничиваться синтаксическим совпадением фраз, а выявлять глубокие семантические соответствия, включая синонимы, перефразировки и контекстуальные намёки.

Для этого используется архитектура на основе эмбеддингов предложений, полученных с помощью модели SentenceTransformer. Сравнение осуществляется по косинусному сходству между эмбеддингами.

### **Выбор модели и обоснование архитектурного решения**

Для генерации эмбеддингов была выбрана модель **sergeyzh/LaBSE-ru-turbo** [8], основанная на LaBSE (Language-agnostic BERT Sentence Embedding). Данная модель ориентирована на мультиязычные данные и показывает хорошие результаты на задачах поиска и кластеризации текстов.

Причины выбора:

* **Поддержка русского языка** и многоязычных текстов;
* Архитектура **bi-encoder**, обеспечивающая высокую скорость при поиске схожих текстов;
* Высокие показатели на **MTEB-бенчмарке**, особенно для задач, связанных с информационным поиском [9];
* Возможность дообучения с помощью библиотеки sentence-transformers.

### **Алгоритм обучения и функция потерь**

Для обучения использовалась функция **MultipleNegativesRankingLoss**. Её особенность в том, что она позволяет обучать модель на положительных парах (вакансия — релевантное резюме), не требуя явного задания отрицательных примеров. Остальные пары в батче автоматически считаются отрицательными, что повышает обобщающую способность модели и особенно полезно при ограниченном объёме размеченных данных.

### **Подготовка обучающих данных**

Формирование обучающей выборки основывалось на корпусе текстов, полученных из реальных вакансий и резюме, собранных с открытых источников интернет-рекрутмента. Эти данные отражают актуальные требования работодателей и реальные профили соискателей, что позволило учесть особенности живого делового языка, структуры описаний и типичных формулировок.

Для увеличения объёма обучающего материала и повышения устойчивости модели к лексическим и синтаксическим вариациям часть обучающих пар была сгенерирована с использованием нейросетевых методов.

Таким образом, итоговая выборка включает как данные, отражающие реальные профессиональные контексты, так и автоматически расширенные вариации, что позволило получить достаточно разнообразный и сбалансированный набор положительных примеров для эффективной донастройки модели.

# Архитектура и реализация веб-приложения

Данная глава описывает архитектуру веб-приложения, его ключевые компоненты, внутреннюю структуру, используемые технологии и способы интеграции модели машинного обучения в общую систему. Приложение реализовано с использованием клиент-серверной архитектуры, где фронтенд и бэкенд взаимодействуют через REST API.

### **Общая архитектура и логика взаимодействия компонентов**

Веб-приложение построено по принципу разделения на клиентскую и серверную части. Клиентская часть отвечает за отображение пользовательского интерфейса и отправку запросов к серверу, а серверная часть обрабатывает эти запросы, работает с базой данных и вызывает ML-модель для оценки соответствия кандидатов вакансиям.

Взаимодействие компонентов происходит по следующей схеме:

* Пользователь вводит данные (резюме или вакансию) через HTML-форму.
* Клиентская часть отправляет данные в формате JSON на сервер по API.
* Сервер сохраняет данные в базе и, при необходимости, вызывает ML-модуль для обработки.
* Результаты сравнения (рейтинги релевантности) отправляются обратно клиенту и отображаются в интерфейсе.

Такой подход позволяет добиться масштабируемости, изолировать логику сравнения от пользовательского интерфейса и упростить поддержку.

Для более детального понимания логики взаимодействия пользователей с системой приведем описание ролей пользователей и вариантов их использования. В таблице ниже представлены основные сценарии взаимодействия пользователей с системой, а также роли, которые они выполняют в процессе работы приложения.

Таблица 1- Акторы системы

| Актор | Краткое описание |
| --- | --- |
| Гость | Неавторизованный пользователь системы. |
| Администратор | Авторизованный пользователь с правами доступа, может быть добавлен через панель администратора или специальной командой в терминале Django. |
| Система | Исполняет действия и взаимодействует с БД/ML-моделью, может взаимодействовать с пользователями через API. |

Таблица 2- Варианты использования

| Основной актор | Вариант использования | Формулировка |
| --- | --- | --- |
| Администратор | Авторизоваться | Пройти проверку корректности пароля и логина. |
| Добавить резюме/вакансию | Заполнить корректно поля соответствующей формы данными и нажать на кнопку отправить. Отправить данные в формате JSON через API. |
| Просмотреть список резюме/вакансий | В случае наличия данных в БД Система покажет 5 последних записей резюме/вакансий. |
| Запустить сопоставление резюме с вакансией | Выбирается вакансия, система запускает ML-анализ, выводится список наиболее релевантных резюме. |
| Управлять записями через админ-панель | Полный CRUD по моделям через стандартный интерфейс Django. |
| Гость | Отправить резюме | Заполнить корректно поля формы данными и нажать на кнопку отправить. Отправить данные в формате JSON через API. |
| Система | Сохранить данные | Сохранить полученные данные в БД. |
| Вызвать ML-модуль | Запустить локально алгоритм, использующий SentenceTransformer. |
| Рассчитать оценку соответствия | Привести полученные значения от ML модуля к удобному и отсортировать имеющиеся записи в порядке убывания релевантности. |
| Вернуть информацию из БД | Предоставить пользователю в требуемом формате данные, полученные из БД. |

Также на основе этих вариантов использования построена диаграмма вариантов использования (Use Case Diagram), отражающая ключевые взаимодействия пользователей с системой.

Изображение выглядит как текст, диаграмма, круг, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3- Диаграмма вариантов использования

### **Серверная часть: Django и структура API**

Серверная часть реализована с использованием фреймворка **Django** и его расширения **Django REST Framework** для создания RESTful API. Сервер обрабатывает запросы по следующим основным маршрутам:

* POST /api/resumes/ — добавление резюме;
* POST /api/vacancies/ — добавление вакансии;
* GET /api/resumes/ и GET /api/vacancies/ — получение последних записей;
* GET /api/match/<vacancy\_id>/ — получение списка наиболее подходящих кандидатов по вакансии.

Каждый объект резюме и вакансии сериализуется в JSON. Система использует авторизацию по токену, чтобы ограничить доступ к данным.

### **Клиентская часть: HTML/JS и пользовательские сценарии**

Клиентская часть построена без фреймворков, на чистом HTML и JavaScript (Vanilla JS). Основные элементы:

* Формы добавления резюме и вакансий;
* Выпадающий список для выбора вакансии;
* Кнопка запроса подходящих кандидатов;
* Динамически обновляемый список с результатами.

JS-код отправляет запросы через fetch, обрабатывает ответы и вставляет данные в DOM. Добавлены визуальные улучшения: выделение резюме, выравнивание процентов по правому краю, отображение контактной информации и границы между результатами.

### **Модель данных и структура базы (SQLite)**

На этапе разработки в качестве базы данных используется SQLite благодаря её простоте и автономности.

Структура базы данных приложения включает три основные бизнес-модели:

* Resume (ФИО, телефон, email, текст резюме, желаемая зарплата);
* Vacancy (название, описание);
* Match (ID резюме, ID вакансии, оценка соответствия, текст резюме, созданный результатом ML).

Дополнительно в проекте используется модель CustomUser, реализующая аутентификацию пользователей. Она поддерживается Django через служебные таблицы (например, таблицы пользователей, групп, сессий), которые создаются автоматически во время миграций и необходимы для корректной работы приложения.

Изображение выглядит как текст, снимок экрана, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 - Скриншот из программы Pycharm с перечислением всех таблиц проекта

Для наглядности в ERD отражены только ключевые бизнес-модели, участвующие в сопоставлении резюме и вакансий. Это позволяет сосредоточиться на предметной области и не перегружать диаграмму деталями технической реализации.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 - ERD, отражающая основные бизнес-модели

### **Интеграция ML-модуля с веб-приложением**

Ключевая особенность приложения — использование алгоритма семантического сопоставления текста на базе **SentenceTransformer** (модель sergeyzh/LaBSE-ru-turbo).

Порядок работы модуля:

1. При вызове /api/match/<vacancy\_id>/ сервер получает текст вакансии и сравнивает его с текстами всех резюме в базе.
2. Для этого вызывается Python-функция, в которой:
   1. оба текста кодируются в эмбеддинги через SentenceTransformer;
   2. рассчитывается **косинусное сходство**;
   3. выбираются топ-5 наиболее релевантных резюме.
3. Результаты сериализуются и отправляются клиенту.

Таким образом, модель интегрирована как часть серверной логики и не требует выделенного микросервиса. Это упрощает развертывание и снижает техническую сложность на этапе дипломной разработки.

### **Разработанные прототипы интерфейса**

Для удобного взаимодействия с системой было разработано три основных прототипа веб-страниц:

#### **Главная страница (домашняя)**

Доступна всем пользователям без авторизации. Служит точкой входа для пользователей, которые хотят отправить своё резюме на рассмотрение. На странице предусмотрена форма для загрузки текста резюме, а также кнопка отправки. Такой упрощённый интерфейс позволяет быстро и удобно передать информацию в систему, не отвлекая пользователя лишними деталями.

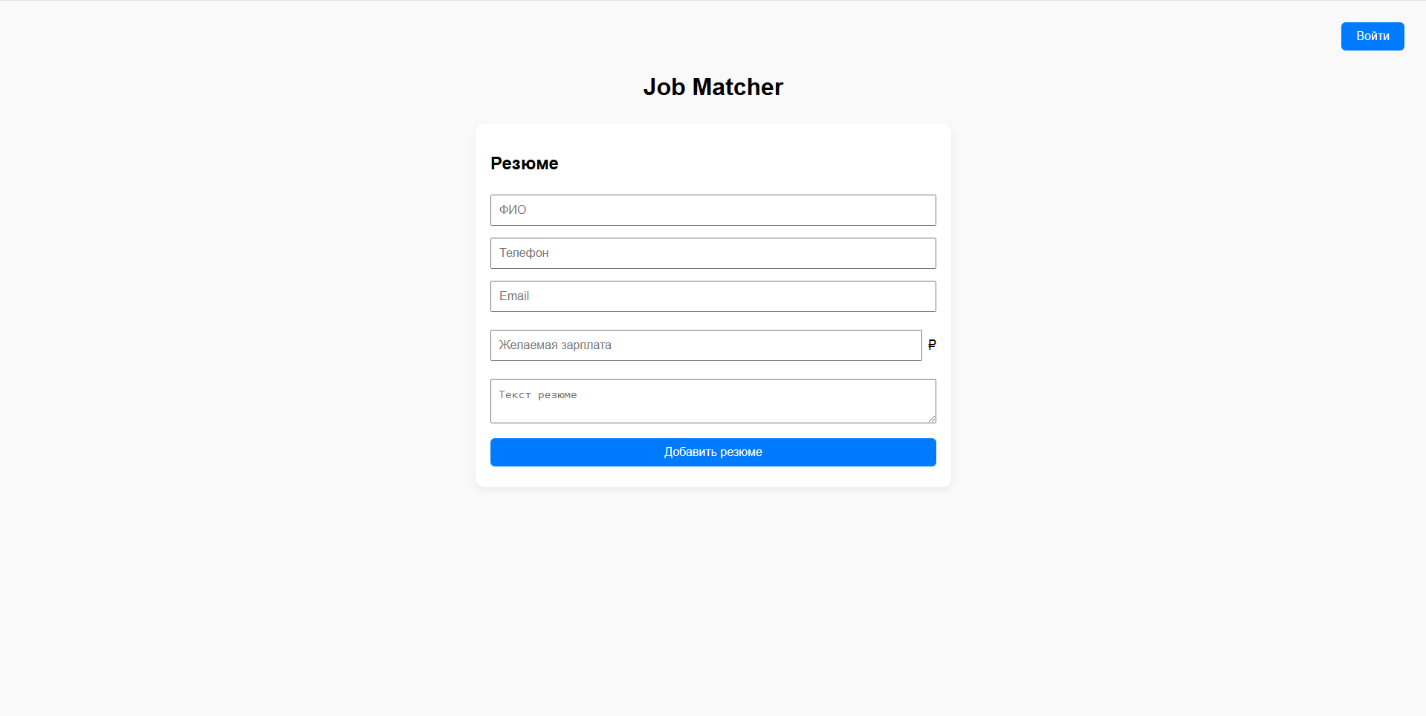


Рисунок 6 - Прототип интерфейса (домашняя страница)

#### **Страница авторизации (логин)**

Предназначена для входа зарегистрированных пользователей (администратора). На странице реализована форма для ввода адреса электронной почты и пароля. Авторизация позволяет администратору получить доступ к функциям управления системой, включая добавление новых вакансий, загрузку резюме и просмотр результатов ранжирования.

Изображение выглядит как снимок экрана, текст, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7- Прототип интерфейса (страница логин)

#### **Административная страница**

Доступна только авторизованному администратору. На этой странице реализованы функции добавления новых вакансий и резюме, а также возможность просмотра совпадений между вакансиями и кандидатами. Страница выполнена в удобном и лаконичном стиле, чтобы администратор мог быстро управлять данными и получать необходимые результаты.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 - Прототип интерфейса (страница администратора)

Таким образом, каждый из прототипов отвечает своей задаче: приём данных, авторизация и управление системой, что обеспечивает удобство и понятность интерфейса для разных категорий пользователей.

# Функциональные возможности системы

Веб-приложение для подбора кандидатов включает в себя ряд функциональных компонентов, разделённых по ролям доступа. В системе предусмотрены две основные категории пользователей: *внешние кандидаты* (неавторизованные) и *администратор* (авторизованный пользователь). В зависимости от роли доступны различные интерфейсы и действия.

### **Авторизация и разграничение доступа**

Вход в административную панель осуществляется через страницу авторизации. После успешного входа пользователь получает токен доступа, который сохраняется в локальном хранилище браузера и используется при обращении к защищённым API-эндпоинтам.

Система реализует разграничение доступа по следующему принципу:

* **Гостевой пользователь** (не авторизован): доступ только к форме отправки резюме.
* **Администратор** (авторизован): доступ к управлению вакансиями и резюме, а также к системе сопоставления кандидатов.

Это обеспечивает безопасность данных и предотвращает несанкционированный доступ к административной информации.

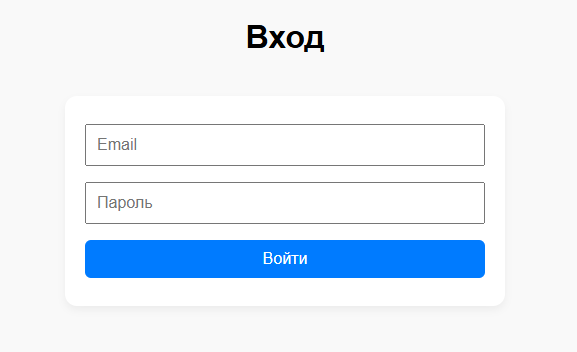
**

Рисунок 9 - Форма для авторизации и получения доступа к панели управления

### **Интерфейс отправки резюме (для неавторизованных пользователей)**

Публичная страница содержит простую и интуитивно понятную форму для загрузки резюме. Пользователь может ввести следующие данные:

* ФИО;
* номер телефона или email;
* Желаемая зарплата;
* текст резюме в свободной форме.

После отправки данные передаются на сервер и сохраняются в базе данных. Пользователь не получает доступ к просмотру других резюме или вакансий.

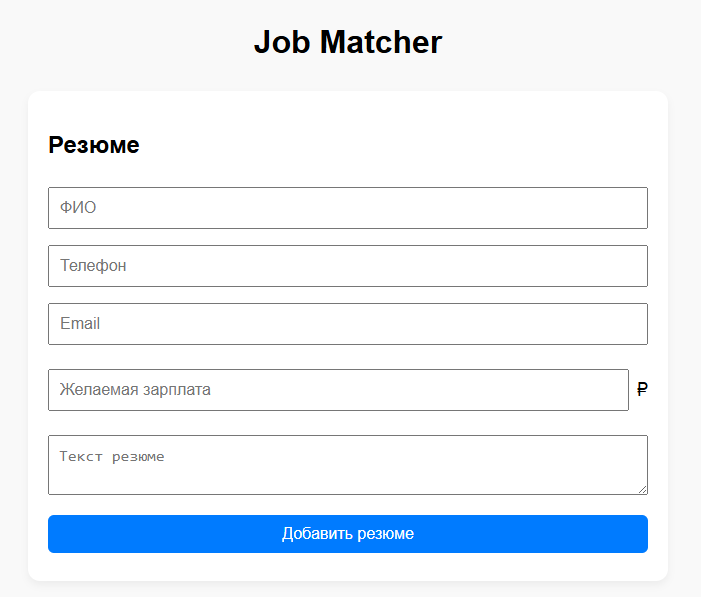


Рисунок 10 - Форма добавления резюме у неавторизованных пользователей

### **Интерфейс администратора: управление вакансиями и резюме**

После авторизации администратор получает доступ к панели управления, содержащей три основные секции:

1. **Список и форма добавления резюме**

Позволяет просматривать последние резюме и добавлять новые вручную. Используется та же структура, что и на публичной странице, но с возможностью просмотра последних добавленных резюме.

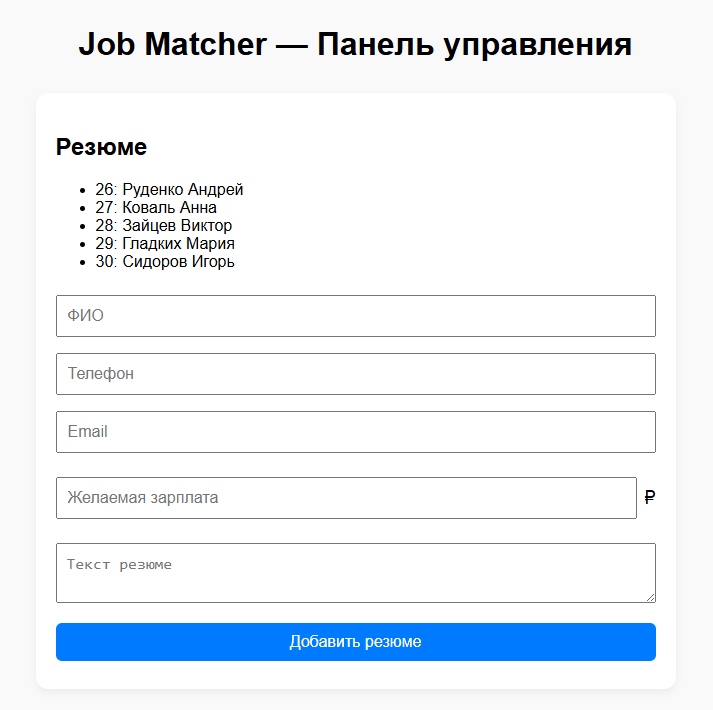


Рисунок 11 - Форма добавления резюме у администратора

1. **Список и форма добавления вакансий**

Администратор может создать описание вакансии, указав её название и требования в свободной форме. Сохраняется текст вакансии для дальнейшего сопоставления с резюме.

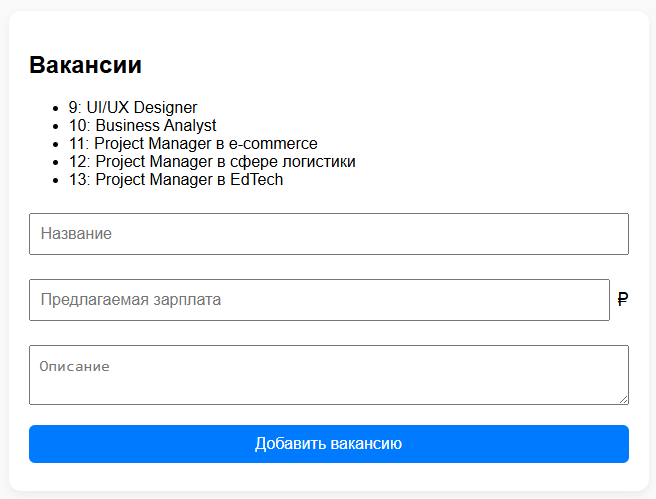


Рисунок 12 - Форма добавления вакансии

1. **Механизм сопоставления кандидатов с вакансиями**

На основе выбранной вакансии запускается процесс ранжирования. Отображаются до 5 наиболее подходящих резюме с их описанием, оценкой соответствия и контактными данными.

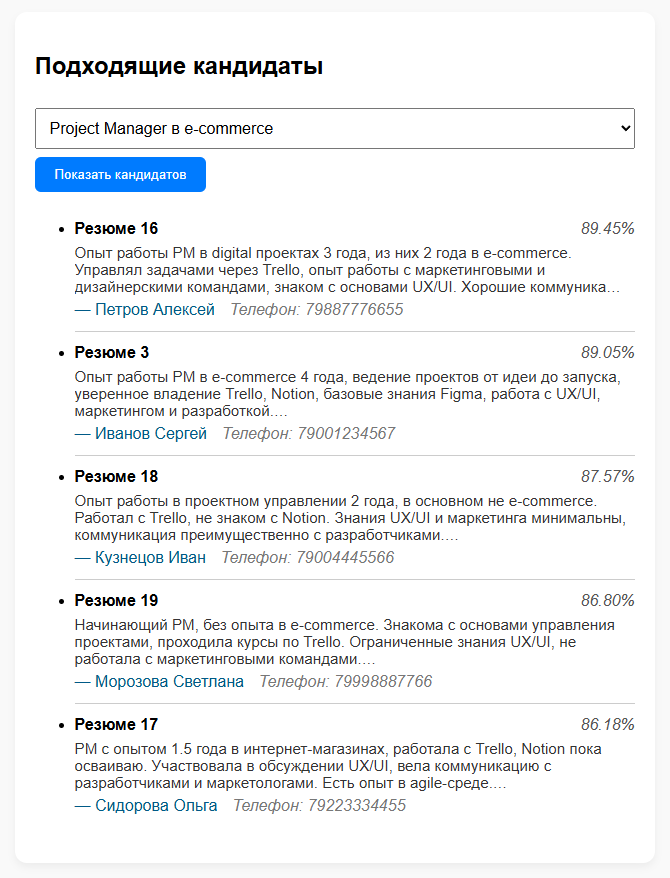


Рисунок 13 - Форма для подбора подходящих кандидатов с результатом выдачи

### **Встроенная административная панель Django**

Помимо разработанного пользовательского интерфейса, система также использует стандартную административную панель Django, которая предоставляет удобные средства для управления данными через веб-интерфейс.

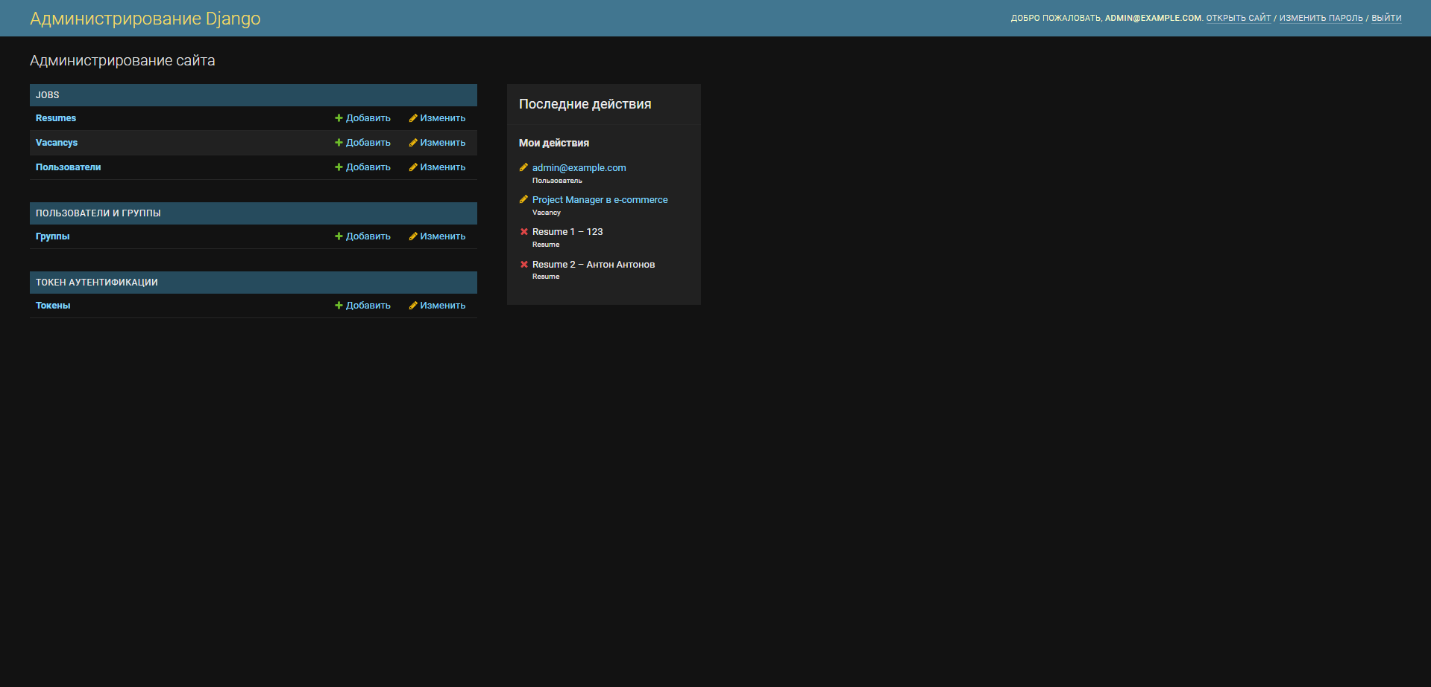


Рисунок 14 - Страница стандартной административной панели Django

Основные возможности админ-панели:

* *Аутентификация и контроль доступа*

Доступ к административной панели предоставляется только зарегистрированным пользователям с соответствующими правами. Используется встроенная система управления пользователями Django.

* *Управление сущностями (моделями)*

Через интерфейс админки можно:

* + просматривать, редактировать и удалять записи в базе данных;
  + фильтровать и сортировать резюме и вакансии;
  + быстро находить нужные объекты с помощью встроенного поиска.
* *Редактирование на месте*

Возможность напрямую изменять данные без необходимости заходить в каждый объект — например, редактировать ФИО или зарплату резюме из списка.

* *Автоматическая генерация форм*

Django автоматически создаёт формы редактирования на основе моделей. Это упрощает разработку и обеспечивает единообразие интерфейса.

* *Расширяемость*

Админ-панель может быть расширена кастомными представлениями, дополнительными действиями и пользовательскими фильтрами при необходимости.

Админка Django выполняет вспомогательную функцию и удобна для внутреннего администрирования и отладки. В отличие от пользовательской панели администратора, реализованной вручную в проекте, встроенная панель предоставляет полный доступ ко всем данным в системе, в том числе к объектам, которые не отражаются в основном интерфейсе (например, пользователи, токены).

Это решение существенно упростило разработку и управление системой на этапе отладки и остаётся полезным инструментом для администраторов и разработчиков в процессе сопровождения проекта.

# Реализация и тестирование

В этой главе описаны этапы разработки системы, методика тестирования, а также представлены основы анализа результатов работы алгоритма подбора кандидатов. Основное внимание уделено проверке корректности логики, стабильности работы веб-приложения и точности ML-модуля.

### **Этапы разработки и технологии реализации**

Разработка системы велась итеративно, с последовательной реализацией базовой функциональности и последующим расширением.

*Основные этапы:*

* проектирование архитектуры клиент-серверного приложения;
* разработка REST API на основе Django;
* реализация интерфейсов с использованием HTML и JavaScript;
* построение модели сравнения на базе SentenceTransformer;
* обучение модели на парных выборках вакансий и резюме;
* интеграция ML-модуля с веб-приложением;
* проведение тестирования и валидации.

*Использованные технологии:*

* Back-end: Django + Django REST Framework;
* Front-end: HTML, CSS, нативный JavaScript;
* База данных: SQLite;
* ML: HuggingFace Transformers, модель sergeyzh/LaBSE-ru-turbo, дообученная с использованием MultipleNegativesRankingLoss;
* Среда разработки: PyCharm.

### **Тестирования алгоритма подбора кандидатов**

Для оценки корректности и эффективности алгоритма семантического сопоставления вакансий и резюме было проведено локальное тестирование с использованием заранее подготовленных описаний вакансий и резюме кандидатов. Цель теста заключалась в том, чтобы проверить способность модели правильно различать степень релевантности кандидатов на основе их опыта и соответствия требованиям вакансии.

#### **Методика тестирования**

* *Сравнение с кандидатами*

Алгоритм последовательно сравнивает вакансию с рядом описаний кандидатов, различающихся по уровню опыта и полноте соответствия заявленным требованиям. Для каждой пары «вакансия — резюме» вычисляется косинусное сходство, преобразованное в процент совпадения.

* *Сравнение с ручной оценкой*

Для проверки корректности результатов автоматической оценки используется ручное ранжирование кандидатов. Это позволяет провести визуальное и аналитическое сравнение ожидаемого порядка с результатами, выданными моделью.

#### **Описание тестовой вакансии и групп кандидатов**

В качестве тестовой была выбрана типовая вакансия Project Manager в продуктовую IT-компанию, разрабатывающую финансовые веб-приложения. Описание вакансии включало как общие управленческие требования (ведение Scrum-команд, контроль сроков, управление бэклогом), так и специфические технические навыки (знание Jira, Confluence, опыт с API, понимание TDD/CI/CD, высокий уровень английского языка). Это позволило обеспечить баланс между универсальными и специализированными критериями отбора.

Для оценки работы алгоритма были сформированы три группы кандидатов с заранее заданными описаниями:

* **Группа 1: Реалистичные описания с «водой»**

В эту группу вошли пять резюме, приближённых к реальным, с разной степенью соответствия вакансии. Описания кандидатов содержали субъективные формулировки, неполные сведения, а также "воду" — фразы общего характера без конкретики. Целью было проверить, сможет ли модель выделить действительно релевантные профили в условиях, приближенных к реальному потоку откликов, где резюме могут быть неполными, перегруженными эмоциями или недостаточно структурированными.

* **Группа 2: Смешанные кандидаты (разная релевантность)**

В эту группу включены резюме как релевантных специалистов (например, Project Manager с опытом в финансовом IT-секторе), так и нерелевантных (например, садовод без технического бэкграунда). Также представлены промежуточные случаи — специалисты из смежных областей, таких как DevOps или бизнес-анализ. Эта группа позволила протестировать чувствительность модели к очевидным и менее очевидным различиям между кандидатами.

* **Группа 3: Реалистичные + нерелевантные кандидаты**

В эту группу включены 5 резюме из 1 группы и 3 нерелевантных резюме. Целью было проверить, сможет ли модель выделить всех релевантных кандидатов.

#### **Выбранные метрики**

Для количественного анализа были рассчитаны следующие метрики:

* *Recall@5* — это показатель, который отражает долю релевантных кандидатов, попавших в топ-5 результатов, относительно общего количества релевантных кандидатов в выборке (6.1).

где:

*Rel* - множество релевантных кандидатов (по экспертной оценке);

*Top5* - множество кандидатов, ранжированных моделью в топ-5.

* *Precision@1* — это показатель, который отвечает на вопрос: «Сколько раз в тестах модель поставила релевантного кандидата на первое место в ранжировании?» (6.2). Эта метрика демонстрирует способность модели правильно выделять наиболее релевантного кандидата.
* *Spearman’s rho* — это коэффициент ранговой корреляции, который измеряет, насколько хорошо модель сохранила порядок кандидатов в сравнении с экспертным эталоном (6.3). Значение 1.0 означает полное совпадение порядков, 0.0 — отсутствие взаимосвязи, а -1.0 — полностью обратный порядок.

где

— количество кандидатов в сравнении (например, топ-5);

— разность рангов для i-го кандидата между моделью и экспертным эталоном.

### **Оценка результатов тестирования**

Для количественной оценки качества сопоставления использовались три тестовые группы кандидатов с заранее определённой релевантностью.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 - Результат тестирование на 1 и 2 группах

Во второй группе (смешанные кандидаты), где представлены как релевантные, так и нерелевантные профили, модель практически полностью совпала с ожидаемым порядком ранжирования: наиболее подходящий кандидат (ID 8) занял первое место с результатом 91.38% совпадения, а наименее релевантный (ID 6) оказался на последнем месте с 69.85%. Алгоритм показал 100% точность попадания релевантного кандидата на первое место (Precision@1 = 1.00) и полную согласованность ранжирования с эталоном (Spearman’s rho = 1.00).

В первой группе (реалистичные и размытые описания кандидатов) наблюдались незначительные расхождения с ожидаемым порядком: кандидат с ID 1 получил оценку 83.70% и оказался на втором месте, опередив кандидата с ID 5 (83.35%), хотя кандидат 5 был более релевантен. Эти отклонения можно объяснить субъективными формулировками в резюме (например, «люблю общаться», «предпочитаю договоренности») и менее структурированным стилем текста, который влияет на векторное представление и ранжирование. Несмотря на это, модель показала Precision@1 = 1.00, а Spearman’s rho = 0.80, что свидетельствует о высоком качестве ранжирования.

В третьей группе (реалистичные описания плюс нерелевантные кандидаты) модель продемонстрировала высокую чувствительность: все релевантные кандидаты попали в топ-5 (Recall@5 = 1.00), а наиболее релевантный кандидат занял первое место (Precision@1 = 1.00). Spearman’s rho составил 0.80, что указывает на хорошее согласование порядка ранжирования модели с эталоном, даже с учётом дополнительных нерелевантных кандидатов.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16 - Результат тестирование на 3-ей группе

Эти результаты демонстрируют, что модель SentenceTransformer способна эффективно выделять наиболее релевантных кандидатов и правильно их ранжировать, особенно в случаях с чётко выраженными критериями вакансии. При работе с «размытыми» резюме также наблюдается высокая чувствительность к релевантным профилям, несмотря на небольшие отклонения. В целом, модель обеспечивает надёжное сопоставление резюме и вакансий, что особенно важно для задач автоматизированного подбора персонала.

# Экономическое обоснование

### **Концепция**

Целью данной ВКР является разработка веб-приложения для автоматизированного подбора подходящих кандидатов на основе анализа их резюме и текста вакансий. Приложение построено по клиент-серверной архитектуре с использованием технологий HTML, JavaScript (vanilla) и фреймворка Django на стороне сервера. Данные хранятся в базе данных SQLite. Ключевым элементом является алгоритм семантического сравнения текста вакансий и резюме, реализованный с использованием модели SentenceTransformer.

В рамках ВКР были реализованы:

* загрузка вакансий и резюме через веб-интерфейс;
* отображение списка вакансий;
* ранжирование кандидатов по релевантности вакансии;
* упрощённый пользовательский интерфейс для управления данными.

Для реализации веб-приложения использовались преимущественно бесплатные инструменты и открытые библиотеки:

* фреймворк Django (бесплатный, open-source);
* язык программирования Python (бесплатный, open-source);
* библиотека SentenceTransformer для работы с эмбеддингами (бесплатная, open-source);
* база данных SQLite (бесплатная, встроенная в Python);
* клиентская часть — HTML/JS (vanilla) (бесплатно);
* текстовый редактор PyCharm Community Edition (бесплатные версии);
* система контроля версий Git (бесплатная).

### **Расчет затрат на оплату труда**

Для корректного расчета затраты на этапах проектирования и разработки необходимо сформировать план проводимых работ. На основе разработанного плана будет определена трудоемкость работ и рассчитана ставка заработной платы работников на день. Время, затраченное на выполнение работы студентом, составляет 56 дней; а время, затраченное на выполнение работы научным руководителем - 7 дней.

Таблица 3 - Затраты по этапам работы

| № | Наименование работ | Исполнитель | Длительность работ в днях |
| --- | --- | --- | --- |
| 1 | Постановка целей и задач, выдача технического задания | Руководитель | 2 |
| 2 | Анализ и сравнение существующих систем | Студент | 7 |
| 3 | Поиск существующих алгоритмов формирования рекомендаций и их реализация | Студент | 5 |
| 4 | Разработка архитектуры приложения | Студент | 5 |
| 5 | Реализация интерфейса и логики веб-приложения | Студент | 15 |
| 6 | Тестирование веб-приложения | Студент | 10 |
| 7 | Оформление пояснительной записки | Студент | 9 |
| 8 | Консультации с научным руководителем | Руководитель | 5 |
| Студент | 5 |

По данным сайта hh.ru, в Санкт-Петербурге и Ленинградской области открыто 6 вакансий по профессии “django developer” уровне стажер или junior. Средняя заработная плата составляет 30 000 рублей [10]. По профессии преподаватель на направлении «Информационные системы и технологии» открыто 56 вакансий, со средней заработной платой в 70 000 рублей [11].

На основе этой информации рассчитываются дневные ставки программиста и преподавателя:

Трудозатраты научного руководителя:

Трудозатраты программиста:

На основе данных о трудоемкости работ и ставок исполнителей, участвующих в проекте, рассчитываются расходы на заработную плату. Процент отчислений на социальные нужды 30%, процент дополнительной заработной платы 12%.

Основная заработная плата рассчитывается по формуле:

где - время, затраченное i-м исполнителем на проведение работ (в днях);

- ставка i-го исполнителя (в руб./день)

Дополнительная заработная плата составляет:

Отчисления на социальные нужды от основной и дополнительной заработной платы равны:

### **Расчет накладных расходов**

В данной работе норматив накладных расходов равен 20% от суммы основной и дополнительной заработной платы. Накладные расходы рассчитываются по формуле:

### **Расходы на материалы**

Данные при расчете количества и стоимости материалов с учетом транспортно-заготовительных расходов занесены в таблицу. К расходам на материалы относят затраты на основные и вспомогательные материалы. Норма транспортно-заготовительных расходов = 10%.

Таблица 4 - Затраты на материалы

| Материалы | Кол-во | Цена, руб. | Сумма, руб. |
| --- | --- | --- | --- |
| Бумага офисная Projecta Special A4, 500 листов, 80г/м2, 104 мкм [12] | 1 | 464 | 464 |
| Комплект картриджей HP (305X) совместимые 4 цвета CE410X / CE411A / CE412A / CE413A для HP LaserJet Pro M351/M451/M475 [13] | 1 | 4062 | 4062 |
| ИТОГО: | | | 4526 |
| Транспортно-заготовительные расходы (10%) | | | 452,6 |
| ВСЕГО: | | | 4978,6 |

### **Издержки на амортизацию ПК и оргтехники**

В процессе разработки приложения использовался ПК и принтер. Стоимость ноутбука MSI GF63 Thin 12UC-1036XRU (9S7-16R821-1036) – 84209 [14], стоимость принтера HP LaserJet M111a (7MD67A) A4 – 9422 [15].

Сумма амортизации за год вычисляется по формуле:

где стоимость оборудования; норма амортизации.

Согласно постановлению правительства РФ от 01.01.2002 №1 (ред. от 28.04.2018) «О Классификации основных средств, включаемых в амортизационные группы» персональные компьютеры и печатающие устройства к ним, относятся ко второй группе. Срок полезного использования оборудования равен от двух до трех лет включительно. Срок эксплуатации три года [16]. Норма амортизации = 33,3%.

руб.

Сумма амортизации за рабочий день равна:

В 2025 году 247 рабочих дней, соответственно:

Амортизация оборудования за время проектирования ВКР:

### **Расходы на услуги сторонних организаций**

Для проектирования ВКР была необходимость доступа в интернет. Поэтому была подключена интернет услуга компании ООО «СкайНэт», стоимость которой составляет 800 руб. в месяц.

Таблица 5- Затраты по работам, выполняемым сторонними организациями

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Кол-во | Цена, руб. | Сумма, руб. |
| Доступ в интернет, мес. | 2 | 800 | 1600 |
| ИТОГО: | | | 1600 |

### **Полная себестоимость работы**

Себестоимость ВКР по статьям расхода и в целом указана в таблице 6.

Таблица 6 - Смета затрат на ВКР

|  |  |  |
| --- | --- | --- |
| Наименование статьи | Сумма (руб.) | Структура себестоимости, (%) |
| Материальные затраты | 4 978,6 | 2,68 |
| Расходы на оплату труда | 115 732,98 | 62,34 |
| Отчисления на социальные нужды | 34 719,89 | 18,70 |
| Амортизационные отчисления | 7 068 | 3,80 |
| Накладные расходы | 23 146,59 | 12,46 |
| Расходы на услуги сторонних организаций | 1 600 | 0,86 |
| Итого | 185 646,06 | 100 |

### **Вывод**

В данном разделе была рассчитана себестоимость ВКР по разработке веб-приложения для подбора кандидатов на основе анализа текстов вакансий и резюме. В результате расчетов получено распределение расходов на оплату труда сотрудников, отчисления на социальные нужды, материальные затраты, амортизацию оборудования, накладные расходы и услуги сторонних организаций.

Наибольшая часть затрат приходится на оплату труда сотрудников — 115 732,98 руб. (62,34% от общей суммы). Значительный удельный вес также занимают отчисления на социальные нужды (18,70%) и накладные расходы (12,46%). Затраты на материалы и амортизацию оборудования составляют относительно небольшую долю — 2,68% и 3,80% соответственно.

Общая себестоимость разработки веб-приложения составляет 185 646,06 руб. Полученные данные позволяют оценить основные направления затрат и подчеркнуть ключевую роль оплаты труда в структуре себестоимости. Для повышения экономической эффективности проекта в дальнейшем может быть рассмотрена возможность оптимизации накладных расходов и привлечение сторонних организаций для выполнения отдельных задач.

# Заключение

***Подведение итогов ВКР***

В ходе выполнения ВКР было разработано веб-приложение для подбора кандидатов на основе семантического анализа текста вакансий и резюме. Система использует дообученную языковую модель SentenceTransformer, позволяющую сопоставлять тексты по смыслу, а не по ключевым словам. Были реализованы пользовательские интерфейсы для добавления вакансий и резюме, а также механизм ранжирования кандидатов по степени релевантности. Результаты тестирования показали, что модель эффективно выделяет подходящих кандидатов, особенно при чётко выраженных различиях в опыте и навыках.

***Практическая значимость полученных результатов***

Разработанная система демонстрирует реальную применимость в задачах предварительного отбора кандидатов. Она может использоваться в составе HR-аналитики или как вспомогательный инструмент для рекрутеров в ИТ-компаниях. Благодаря использованию многоязычной модели и возможности дообучения, система может быть адаптирована под разные профили вакансий и корпоративные стандарты. Предложенное решение также может служить прототипом для более масштабных платформ с автоматическим подбором исполнителей под проектные задачи.

***Направления дальнейших исследований***

Современный рынок труда в ИТ сталкивается с рядом новых вызовов. С одной стороны, растёт объём автоматизированных откликов и некачественных резюме, созданных или отредактированных с помощью ИИ. Кандидаты всё чаще прибегают к искусственному завышению опыта, чтобы пройти автоматические фильтры. С другой стороны, работодатели внедряют боты и шаблонные сценарии взаимодействия, что снижает качество обратной связи и перегружает соискателей бессмысленными коммуникациями.

Дальнейшие исследования могут быть направлены на:

* **Учет достоверных достижений**: интеграция данных о реальных проектах кандидата, таких как репозитории, портфолио, сертификаты и отзывы, позволит уменьшить зависимость от формального описания опыта;
* **Оценку индивидуальности**: развитие метрик, учитывающих не только опыт, но и потенциал, мотивацию и соответствие культуре компании;
* **Борьбу со спамом и накрутками**: внедрение механизмов валидации информации и выявления неаутентичных резюме;
* **Этичное использование ИИ в подборе**: создание прозрачных моделей, которые не только ранжируют, но и объясняют причину рекомендаций, поддерживая доверие со стороны кандидатов.

Таким образом, задача построения эффективной рекомендательной системы выходит за рамки технической реализации и требует комплексного подхода — с учётом социальных, этических и поведенческих аспектов современного рынка труда.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kumar D., Grosz T., Rekabsaz N., Greif E., Schedl M. Fairness of recommender systems in the recruitment domain: an analysis from technical and legal perspectives [Текст] / Kumar D., Grosz T., Rekabsaz N., Greif E., Schedl M. // Front. Big Data. — 2023.
2. Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni, V Adithya Krishnan Applied Recommender Systems with Python [Текст] / Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni, V Adithya Krishnan — 1. — Apress Berkeley, CA, 2022 — 248 c.
3. G. Adomavicius, Alexander Tuzhilin Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions [Текст] / G. Adomavicius, Alexander Tuzhilin // IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. — 2005. — № 17. — С. 734-749.
4. Исследование [Электронный ресурс] URL: <https://www.pwc.com/gx/en/issues/artificial-intelligence/job-barometer/report.pdf> (дата обращения: 21.05.25)
5. Исследование [Электронный ресурс] URL: <https://cognition.ai/blog/swe-bench-technical-report> (дата обращения: 21.05.25)
6. Исследование [Электронный ресурс] URL: <https://mts-link.ru/blog/kolichestvo-vakansij-s-upominaniem-ii-vyroslo-v-dva-raza/?ysclid=m9s8zvvo6h678815170> (дата обращения: 21.05.25)
7. Исследование [Электронный ресурс] URL: <https://spb.hh.ru/article/research?ysclid=m9sak49tmc260801720> (дата обращения: 21.05.25)
8. Модель ИИ на платформе “Hugging Face” [Электронный ресурс] URL: <https://huggingface.co/sergeyzh/LaBSE-ru-turbo/> (дата обращения: 20.05.25)
9. Рейтинг на платформе “Hugging Face” [Электронный ресурс] URL: <https://huggingface.co/spaces/mteb/leaderboard> (дата обращения: 20.05.25)
10. Сайт вакансий hh.ru [Электронный ресурс] URL: https://spb.hh.ru/search/vacancy?text=django+developer&salary=&ored\_clusters=true&experience=noExperience&area=2&hhtmFrom=vacancy\_search\_list&hhtmFromLabel=vacancy\_search\_line (дата посещения: 29.05.2025)
11. Сайт вакансий hh.ru [Электронный ресурс] URL: <https://spb.hh.ru/vacancies/prepodavatel-informatsionnyh-tehnologij> (дата посещения: 29.05.2025)
12. Платформа Яндекс.Маркет, [Электронный ресурс] URL : <https://market.yandex.ru/card/bumaga-ofisnaya-projecta-special-a4-500-listov-80gm2-104-mkm/4401779086?do-waremd5=2qgAr_HCIW3SnXBN49FkWw&sponsored=1&cpc=91jLStYA1po61VdmhAJp4NxlBH5ezG4EFRC0iiYyqDql4aENPN2xRyyL3r4Sy6eL8EPDiWm4bER-oiHqJik9y2xuZtfAO0JtBzp9E6EMXh29_KQFbQ9Jb3bWO4BddVnGS9_TjSQG8ux-BDEY4brHQVSF52Sp9wUpwtg9bD-IrFXL-vBLzwh5T27zWKV6Lg2VyvM9F_6_BqKvhQmheLoaLKBd6O3hi6uZM7UYTf3nBYTn5Yo_qQSbEcSCaZ4ksPANdRtAm0VPVgaRhaWmEPl0vc2smAEIScBYQl5K_fUTppWvhEpNmvAh-G0Xjbx5WyfW3ZpkgzgvlOAXYwTBEpYtBKtwH3EK1F8cmlqisTaAcNqv8kv2hyTep1eiFJF1ittO7GPgLX06abdqD0fZsFQwWuZk70bj7R-ELgJZFQnl6AXVcEKYy1e2Z9yzfAxZgmbMuNnjaTCqvDs65SIBFDNAKLD23wzPxIpiJotzlw1XWQnJmhdx4P4Gsy1YDoufStE8SYUIqgUG-zDE1a-EpGsojgGC61fyK5H3i_3PZT19Vp3qD9HWEDvDIZzMu37WGPHTxIgob_XujD4Sa7OJrJ7JNzTGZXUYeya8HQ7eg80bv9Rsd3txCCjaOCeAMWd7Xl6X66jL4ia048XRK7sJ1V2jNgdROZfotTnn> (дата посещения: 29.05.2025)
13. Платформа Яндекс.Маркет, [Электронный ресурс] URL : <https://market.yandex.ru/card/komplekt-kartridzhey-hp-305x-sovmestimyye-4-tsveta-ce410x--ce411a--ce412a--ce413a-dlya-hp-laserjet-pro-m351a--451dn--m451nw--mfp-m475dw-sovmestimyy/103632477166?do-waremd5=A7kay6E7NqEnLNMkgzcLDQ&sponsored=1&cpc=jyXQCOWUkDj5VREY92XQZafK4elcc_-z1k97wMk7_kZABODNsafVBXS3buL9jzolncRqjwWLWqqGiHE8ucpPQMS4fS9OcLrEh7I9aLKPvtTa7B6E6iwTg89qHZcPAeOUVMNbqoZEZhu9UOlQ11F9p8kEX0HatFwJ0IsfgCsrH-X6ddD6vWKqS4dovoDwu3M46ioVLC3oaDmpfSIP9GGfC27dw4vdjAWIe4gRngv1RVwYYK_Ezr-0fYbRX_KEUarK4Y_flu9J8sY%2C> (дата посещения: 29.05.2025)
14. Платформа Яндекс.Маркет, [Электронный ресурс] URL : <https://market.yandex.ru/card/igrovoy-noutbuk-msi-gf63-thin-12uc-1036xru-9s7-16r821-1036/102705941447?do-waremd5=Qdfxpbf8JMmDwoRkBSXghA&sponsored=1&cpc=kkTpgUnPQ5_CO-2E_FqnLafWtg8GobgFOHQZ7TI3hLaOZ0WMYGrp-jRfrZ-3r2pAgywEWF8Da0cg-So-MQcoVc0_sQaz6RiEYDgB4DNLsiPn8J3eB68e0WPX-0-Ij86OhvSVPLppFX8VmqL94jP0EQwNl8zyZdEnpVX6La91yNI48VpFesm-JcVCl9hr48a-nMuGb6n5NTKx2H5lGO_eBgUb6z-KOH6gllEwvufcxQgDMESydEQ_-59Vlor3cBkYRTbai0J8nvjyoZ33fq2gaIFPMz3N9EavthqNBgbB8NJQhef-wI_bMC8doC7qu2SPk5bngHqlT4HPwUft18Jl6SLYnCF6iymDEvUEASlaiIvo676sYoID3JfmVGMooGJxVw_MLneYc0xsGl_mTWnOo8MwmvYgpjrdDNxuGf7TchbgXJGAqB5spi_akbd9m8nDaAnswniyE9MD0JoDwz9w7ff1GvYDPPdeQrZ5FjcQouFJna47u04ZWv_hQPrY3ovzIqCPYMtkeIIiCxOt7TtuQtvl4ahoDJ0JxKBJYG8rxyQ%2C> (дата посещения: 29.05.2025)
15. Платформа Яндекс.Маркет, [Электронный ресурс] URL : <https://market.yandex.ru/card/printer-lazernyy-hp-laserjet-m111a-7md67a-a4/102160468347?do-waremd5=ESw2Zzr6Yb3fWL7YeLwL9w&sponsored=1&cpc=y98CxZqVBQeXHF1f4vycaeyphbWb7-L7KpGGHLzrmhAudlhdmHPm_sPCN1T7TG8aU7NyPth0VF0Fp-QXSbnkQaNgvFPb77Fww_yeXDuG6Mv15_-8oyozF-CKL-WcijDqHuCA71mpdJkfdN-TJBFoVFsUv9ptjrHO4RPWRZRF2J4Hq-oJzDrunrPZPZh3xkJ93f9Qe6pSdRhmsyyi3mmL6Po-zML7dmrIXI4iAj6qxH0rzU7Y_GyfXQEU5okXKBo-eMeIkGPXrJbR09aZjaYDazCUFVdmazP-IBUof86WSwGQAShig_mhTSSw-Q-jNYhmNKNxsEEAl1TUDaRaDfBUoQzye6C-ro9eS7uEUUxPx_oGSekkF0KmzNuo0A1vIccMROKK4qthupIN6tpn2MeQ1VNiPiPsGOA9TUL29vI3QQ9sMDL_obtrW_nkqg7oKcREDZFeHUN5Xs0eYEDLUkV30HGcE930MmTuobQWBTjHrtfmYRNHAB-URs3mkReNYUTkoWEFaZ4_wiGy1yX08Fy_1v6sEcLSUmy47unhUnj0B-bkIvtfbQrfNHffE1P7gv9rte9neVXW0_s%2C&resale_goods=resale_new> (дата посещения: 29.05.2025)
16. О классификации основных средств, включаемых в амортизационные группы: постановление Правительства Рос. Федерации от 01.01.2002 №1 (ред. от 28.04.2018) // Собрание законодательства Российской Федерации

# Приложение A. Python: описание Django моделей

from django.contrib.auth.base\_user import BaseUserManager  
from django.contrib.auth.models import AbstractUser  
from django.db import models  
  
  
class CustomUserManager(BaseUserManager):  
 use\_in\_migrations = True  
  
 def \_create\_user(self, email, password, \*\*extra\_fields):  
 if not email:  
 raise ValueError('Email must be set')  
 email = self.normalize\_email(email)  
 user = self.model(email=email, \*\*extra\_fields)  
 user.set\_password(password)  
 user.save(using=self.\_db)  
 return user  
  
 def create\_user(self, email, password=None, \*\*extra\_fields):  
 extra\_fields.setdefault('is\_staff', False)  
 extra\_fields.setdefault('is\_superuser', False)  
 return self.\_create\_user(email, password, \*\*extra\_fields)  
  
 def create\_superuser(self, email, password=None, \*\*extra\_fields):  
 extra\_fields.setdefault('is\_staff', True)  
 extra\_fields.setdefault('is\_superuser', True)  
 extra\_fields.setdefault('is\_admin', True)  
  
 if not extra\_fields.get('is\_staff'):  
 raise ValueError('Superuser must have is\_staff=True.')  
 if not extra\_fields.get('is\_superuser'):  
 raise ValueError('Superuser must have is\_superuser=True.')  
  
 return self.\_create\_user(email, password, \*\*extra\_fields)  
  
  
class CustomUser(AbstractUser):  
 username = None  
 email = models.EmailField('email address', unique=True)  
  
 full\_name = models.CharField(max\_length=255, default="")  
 phone\_number = models.CharField(max\_length=20, blank=True, null=True)  
 date\_of\_birth = models.DateField(blank=True, null=True)  
 is\_admin = models.BooleanField(default=False)  
  
 USERNAME\_FIELD = 'email'  
 REQUIRED\_FIELDS = ['full\_name']  
  
 objects = CustomUserManager()  
  
 def \_\_str\_\_(self):  
 return self.email  
  
  
class Resume(models.Model):  
 full\_name = models.CharField(max\_length=255, default="")  
 phone\_number = models.CharField(max\_length=20, blank=True, null=True)  
 mail = models.EmailField(blank=True, null=True)  
 text = models.TextField()  
 salary = models.DecimalField(  
 max\_digits=10, decimal\_places=2, blank=True, null=True,  
 help\_text="Желаемая зарплата"  
 )  
  
 def \_\_str\_\_(self):  
 return f"Resume {self.id} – {self.full\_name}"  
  
  
class Vacancy(models.Model):  
 name = models.CharField(max\_length=255)  
 description = models.TextField()  
 salary = models.DecimalField(  
 max\_digits=10, decimal\_places=2, blank=True, null=True,  
 help\_text="Предлагаемая зарплата"  
 )  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Match(models.Model):  
 resume = models.ForeignKey(Resume, on\_delete=models.CASCADE)  
 vacancy = models.ForeignKey(Vacancy, on\_delete=models.CASCADE)  
 score = models.FloatField(help\_text="Оценка соответствия (в процентах)")  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
  
 def \_\_str\_\_(self):  
 return f"Match: Vacancy {self.vacancy\_id} ↔ Resume {self.resume\_id}"

# Приложение Б. Python: Модуль, отвечающий за ML

import sys  
  
import data\_loader  
import embedding as embedding\_module  
  
  
def main():  
 *"""  
 Главная функция для тонкой настройки модели на основе данных резюме и вакансий.  
 Загружает данные, подготавливает пары текста, обучает модель и сохраняет результат.  
 """* print("Запуск тонкой настройки модели с использованием обучающих данных...")  
  
 # Загрузка данных резюме и вакансий  
 resumes\_data = data\_loader.load\_resumes("data/resumes.csv")  
 vacancies\_data = data\_loader.load\_vacancies("data/vacancies.json")  
  
 if not resumes\_data or not vacancies\_data:  
 print("Отсутствуют данные резюме или вакансий для обучения. "  
 "Пожалуйста, убедитесь, что файлы данных присутствуют.")  
 sys.exit(1)  
  
 # Загрузка базовой или ранее обученной модели  
 model = embedding\_module.load\_model(fine\_tuned\_path="models/fine\_tuned\_model")  
  
 # Загрузка обучающих пар из файла  
 ground\_truth\_pairs = data\_loader.load\_ground\_truth("data/train\_pairs.csv")  
  
 # Подготовка текстовых пар для обучения  
 training\_text\_pairs = []  
 for vacancy\_id, resume\_ids in ground\_truth\_pairs.items():  
 # Поиск текста вакансии по ID  
 vacancy\_entry = next((vacancy for vacancy in vacancies\_data if vacancy['id'] == vacancy\_id), None)  
 if not vacancy\_entry:  
 continue  
 vacancy\_text = vacancy\_entry['text']  
  
 for resume\_id in resume\_ids:  
 # Поиск текста резюме по ID  
 resume\_entry = next((resume for resume in resumes\_data if resume['id'] == resume\_id), None)  
 if not resume\_entry:  
 continue  
 resume\_text = resume\_entry['text']  
 training\_text\_pairs.append((vacancy\_text, resume\_text))  
  
 if not training\_text\_pairs:  
 print("Не найдено обучающих пар. Завершение работы.")  
 sys.exit(1)  
  
 # Тонкая настройка модели  
 model = embedding\_module.fine\_tune\_model(  
 model,  
 training\_text\_pairs,  
 epochs=1,  
 output\_path="models/fine\_tuned\_model",  
 batch\_size=16  
 )  
  
 print("Тонкая настройка модели завершена. "  
 "Обученная модель сохранена в папке models/fine\_tuned\_model/")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()