Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

**ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК**

Руководитель образовательной программы

09.03.04 Программная инженерия

И. Н. Обабков .

(подпись) (Ф.И.О.)

«01» июня 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Пояснительная записка

**ТЕМА**

**Формирование требований для автоматизации процесса по актуализации заработной платы сотрудников для компании «Artsofte»**

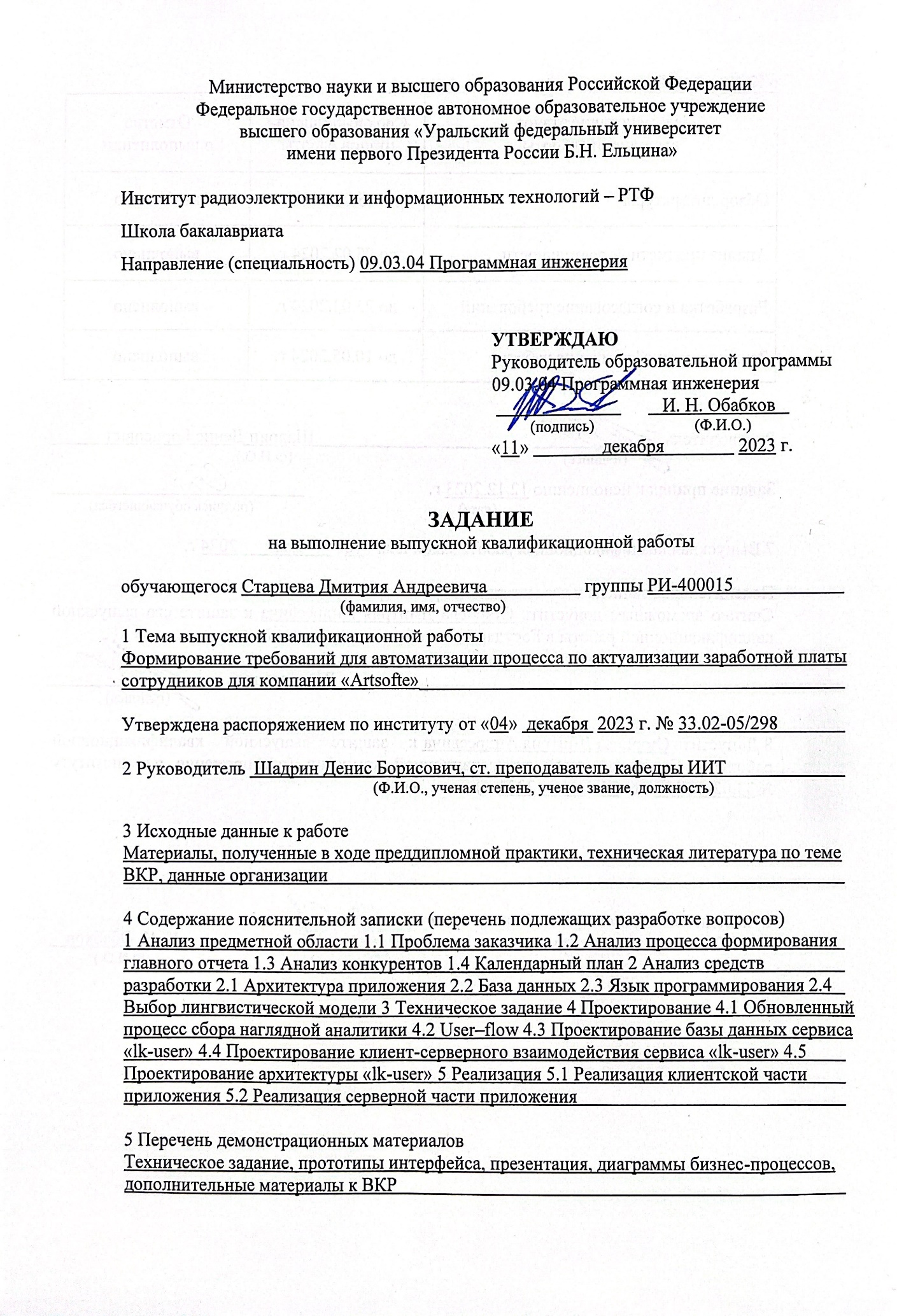
Руководитель: Д.Б. Шадрин \_\_\_\_\_\_\_\_\_\_\_\_\_

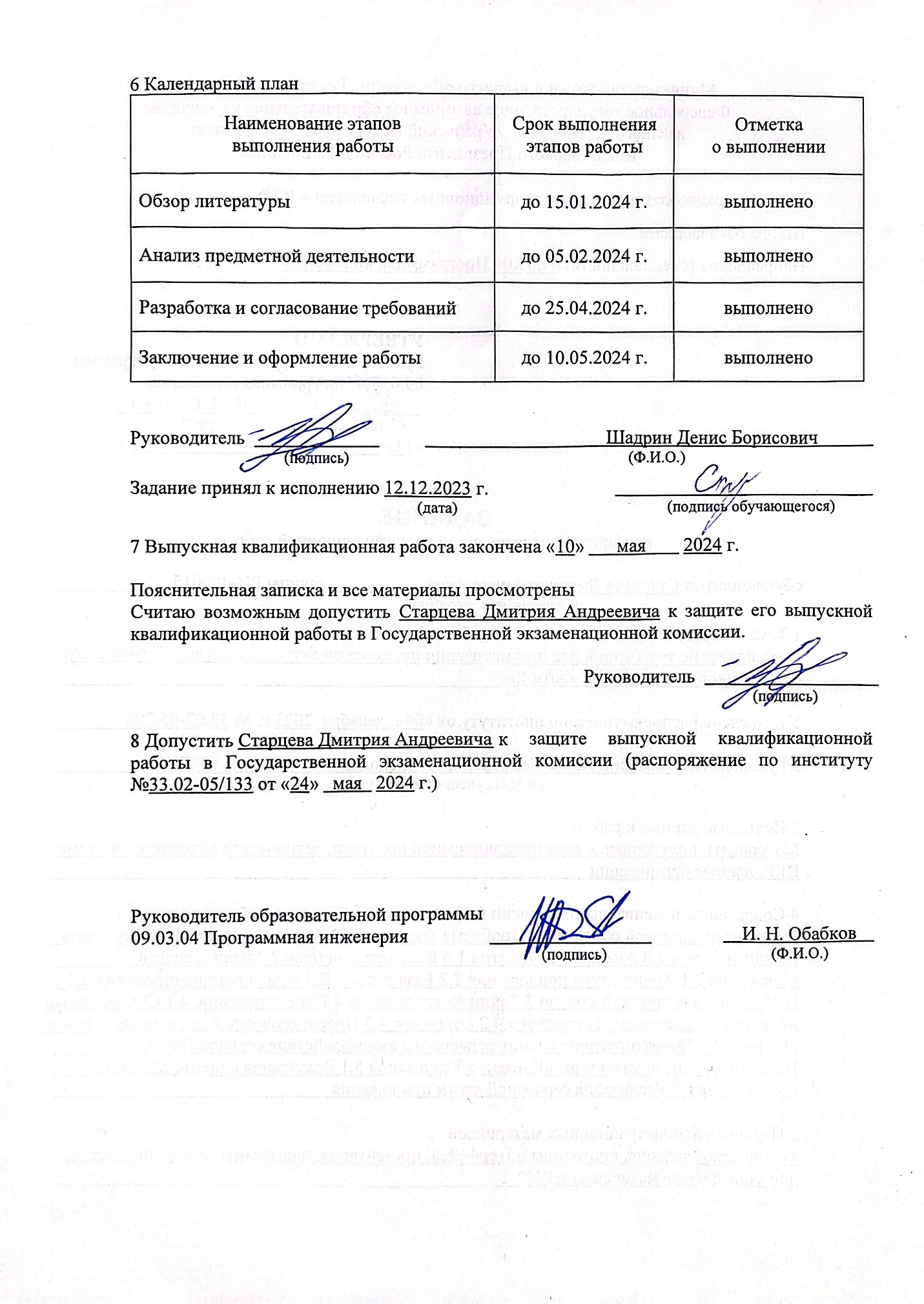
Ст. Преподаватель

Нормоконтролер: В. Н. Васина \_\_\_\_\_\_\_\_\_\_\_\_\_

Обучающийся группы РИ-400015 Д. А. Старцев \_\_\_\_\_\_\_\_\_\_\_\_\_

Екатеринбург 2023



**РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc168030027)

[1 Анализ предметной области 9](#_Toc168030028)

[1.1 Проблема заказчика 9](#_Toc168030029)

[1.2 Анализ процесса формирования главного отчета 11](#_Toc168030030)

[1.3 Анализ конкурентов 15](#_Toc168030031)

[2. Анализ средств разработки 18](#_Toc168030032)

[2.1 Архитектура приложения 18](#_Toc168030033)

[2.2 База данных 20](#_Toc168030034)

[2.3 Язык программирования 23](#_Toc168030035)

[2.4 Выбор лингвистической модели 26](#_Toc168030036)

[3. Техническое задание 30](#_Toc168030037)

[4. Проектирование 34](#_Toc168030038)

[4.1 Обновленный процесс сбора наглядной аналитики 34](#_Toc168030039)

[4.2 User–flow 35](#_Toc168030040)

[4.3 Проектирование базы данных сервиса «lk-user» 38](#_Toc168030041)

[4.4 Проектирование клиент-серверного взаимодействия сервиса 40](#_Toc168030042)

[4.5 Проектирование архитектуры «lk-user» 41](#_Toc168030043)

[5. Реализация 45](#_Toc168030044)

[5.1 Реализация клиентской части приложения 45](#_Toc168030045)

[5.2 Реализация серверной части приложения 47](#_Toc168030046)

[ЗАКЛЮЧЕНИЕ 50](#_Toc168030047)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51](#_Toc168030048)

[ПРИЛОЖЕНИЕ А (обязательное) Схема базы данных сервиса ЛК 53](#_Toc168030049)

[ПРИЛОЖЕНИЕ Б (обязательное) Схема базы данных сервиса аналитики 54](#_Toc168030050)

[ПРИЛОЖЕНИЕ В (обязательное) Техническое задание проекта 55](#_Toc168030051)

[ПРИЛОЖЕНИЕ Г (обязательное) Содержание xlsx файла (отчета) 68](#_Toc168030052)

[ПРИЛОЖЕНИЕ Д (обязательное) Аналитический запрос 72](#_Toc168030053)

[ПРИЛОЖЕНИЕ Е (обязательное) Сущность аналитического запроса 73](#_Toc168030054)

[ПРИЛОЖЕНИЕ Ж (обязательное) Актуализация документов в кодовом представлении 74](#_Toc168030055)

# ВВЕДЕНИЕ

В современном мире цифровизация и автоматизация проникают во все сферы деятельности компаний и повседневной жизни человека. Если раньше технологии касались только технический отделов, то сегодня все нуждается в автоматизации процессов для упрощения работы и повышения ее эффективности. В том числе профессиональная HR–деятельность, без которой не обойдется ни одна компания.

Часто компании прибегают к созданию HRM (Human Resourses Management) отдела, который занимается стратегией обращения с группами сотрудников. Решения HRM должны основываться на определенных данных, сбор которых представляет собой большую работу, которая занимает много ресурсов.

В компании Artsofte В компании работает более 450 человек и оборот кадров всегда является острым вопросом для HR отдела.

Чтобы быть уверенными в своей позиции на рынке кадров: каждый квартал происходит анализ предложения на самом известном сайте по подбору вакансий — hh.ru.

Сбор и систематизация информации о вакансиях на популярном ресурсе hh.ru позволяет не только создать обширную базу данных, но и выделить ключевые тенденции в требованиях работодателей и ожиданиях соискателей. Это обеспечит эффективную адаптацию образовательных программ, обучения кадров и стратегий подбора персонала, что является актуальной задачей в условиях постоянно меняющегося рынка труда.

Чтобы уменьшить нагрузку на HR отдел и эффективно распределить его ресурс – было принято решение разработать сервис для автоматизации процесса актуализации заработной платы сотрудников.

Тем самым целью работы является аналитика процессов организации Artsofte и формирование требований к сервису автоматического анализа рынка – HRParser.

Для достижения цели поставлены следующие задачи:

1. изучение процессов организации встречи в «Artsofte»;
2. выявление и описание требований к системе у заказчика;
3. описание функциональных требование к системе;
4. описание сценариев;
5. передача технического задания в разработку.

Объектом исследования является процесс актуализации заработной платы сотрудников и возможности его автоматизации.

# 1 Анализ предметной области

## 1.1 Проблема заказчика

Перед тем как начать создавать продукт, необходимо провести анализ предметной области. Это включает в себя изучение бизнес–процессов компании, инструментов разработки компании, выявление и изучение проблем, общение с заказчиком, формулирование и проверку гипотез, а также разработку пользовательских историй для лучшего понимания опыта пользователя.

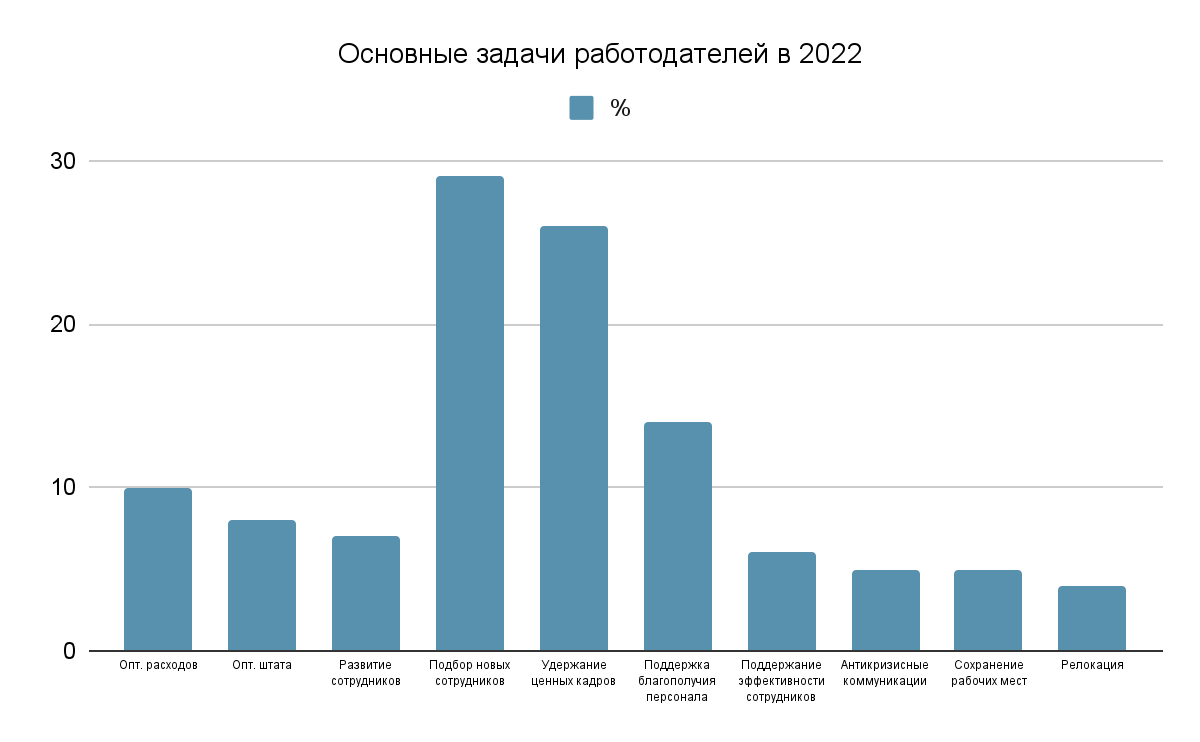
Эта часть работы будет фокусироваться на поиске и изучении проблемы, общении с потенциальными пользователями, и сборе требований. Затем на основе собранной информации будет составлено техническое задание, которое станет основой для разработки архитектуры, выбора инструментов и технологий разработки.

Существует процесс анализа рынка труда с последующей актуализацией заработной платы сотрудников.

Сама актуализация происходит по множественным причинам:

1. актуализация заработной платы позволяет компании прогнозировать будущие расходы на оплату труда. Это важно для предотвращения неожиданных финансовых нагрузок;
2. регулярная актуализация заработной платы позволяет сотрудникам видеть, что их труд и профессиональное развитие оцениваются. Это способствует повышению мотивации, уровня удовлетворенности и приверженности компании;
3. анализ рынка показывает актуальность вакансии: соотношение количества вакансий к заработной плате говорит о востребованности;
4. знание актуальной цены кадра позволяет компании оценить свою конкурентоспособность на рынке труда. А также сравнивать условия труда.

На рисунке 1 приведены главные задачи работодателей в условиях кризиса. График составлен на основе компиляции данных из исследования «TalentTech» в 2022 [1].

Рисунок 1 – Главные задачи работодателей в условиях кризиса по исследованию «TalentTech» в 2022 г.

Относительно этого исследования актуализация заработной платы связана со следующими пунктами:

1. оптимизация расходов;
2. оптимизация штата;
3. подбор новых сотрудников;
4. удержание ценных кадров;
5. поддержание эффективности сотрудников;
6. сохранение рабочих мест.

То есть целевой бизнес–процесс покрывает около 83% задач работодателя в условиях кризиса. Тогда следует чуть ли не обязательное наличие этого процесса в любой IT компании современного времени.

Совсем недавно рост онлайн–активности принес технологическим компаниям исторически высокие прибыли и вызвал бурное увлечение наймом сотрудников, чтобы удовлетворить растущий спрос. В этот период технологические компании предполагали, что такой взлет станет новым стандартом, что привело к активному расширению команд и быстрому увеличению размеров компаний. По статистике из исследования М.Тех [2] За время пандемии Amazon увеличил свою рабочую силу на 93%, Microsoft — на 53%, Meta — на 92%, Apple — на 20%, а Alphabet — на 60%.

У заказчика «Artsofte» имеется 120 ролей и более 450 сотрудников. Здесь под «ролью» понимается функциональная позиция в организации с определенным уровнем квалификации. Текущий бизнес–процесс формирования отчета об актуальности зарплат очень затратный по ресурсам компании. Это тратит время квалифицированного сотрудника. Приходится переносить сроки выполнения остальных задач, в связи с этим. Также если ускорить весь процесс, то компания будет иметь возможность чаще актуализировать заработную плату сотрудникам. Сейчас это происходит раз в полугодие.

Чтобы решить эту проблему нужно найти участок бизнес–процесса, который является самым трудозатраты и автоматизировать работу на этом участке.

## 1.2 Анализ процесса формирования главного отчета

Удобно будет описать процесс формирования итогового отчета с результирующей заработной платой через Customer Touchpoint Map (рисунок 2) – это визуальное представление всех важных моментов взаимодействия клиента с брендом или продуктом на протяжении всего пути от первого знакомства до последующих этапов, целью которой является оптимизация клиентского опыта.

A diagram of a project

Description automatically generated with medium confidence

Рисунок 2 – Customer Touchpoint Map процесса создания итогового отчета

Процесс делится на 5 составляющих:

1. сбор обезличенной аналитики;
2. сбор наглядной аналитики;
3. анализ собранных данных;
4. формирование итоговой презентации;
5. презентация перед руководством.

Сбор обезличенной аналитики – на этом этапе собираются данные по зарплатам из открытых источников. Данные в таких метриках уже структурированы. На данный момент лучшими из бесплатных решений являются:

* https://career.habr.com/salaries,
* https://rit.work/jobs.

Отчеты со сторонних источников, хотя и предоставляют обширные обзоры, вносят свои собственные ограничения в виде ограниченного объема данных и отсутствия возможности проверки их достоверности. Несмотря на эти ограничения, собранный датасет играет важную роль в процессе валидации аналитики на следующем этапе исследования.

На основе анализа этих обезличенных данных мы можем выделить ключевые тенденции и закономерности, которые будут служить основой для более глубоких исследований на следующих этапах. Важно подчеркнуть, что, несмотря на отсутствие возможности проверки конкретных вакансий, полученные данные предоставляют ценный обзор зарплатного уровня в индустрии и являются неотъемлемой частью проверочного этапа нашего исследования.

Стоит отдельно отметить, что hr не контролирует метрики, которые ему предоставляют сторонние ресурсы по анализу рынка, в связи с чем тратятся дополнительные ресурсы на анализ полученных датасетов.

Сбор наглядной аналитики – на этом этапе формируется отчет похожий на тот, что на предыдущем шаге, однако в этом отчете мы можем сами настроить нужные нам метрики, брать только нужные нам данные. Отчет формируется вручную по каждой роли.

По схеме процесса «Сбор данных для наглядной аналитики» (рисунок 3) можно понять, что работа распараллеливается на нескольких человек по причине большого объема ручной работы.

A diagram of a flowchart

Description automatically generated

Рисунок 3 – BPMN диаграмма процесса «Сбор наглядной аналитики»

Работа каждого юнита (Рисунок 3 – Сбор данных в рамках роли) заключается в том, что он заходит на сайт hh.ru, начинает читать вакансии и механически переносить данные со страницы вакансий в excel таблицу – построчно формируя отчет по одной роли. На это уходит большинство времени.

Как можно заметить по рисунку 4 – сбор данных для этапа «Наглядная аналитика #2» занимает около 74% от суммарного времени всех этапов. Именно здесь происходит «утечка» времени. Следовательно, именно этот этап и стоит подвергать автоматизации.

A graph with blue and orange bars

Description automatically generated

Рисунок 4 – Распределение времени среди трёх этапов создания итогового отчёта

Анализ собранных данных представляет собой выявление необходимых для итогового отчета данных:

1. зарплатные ожидания;
2. анализ уровня конкурентоспособности;
3. тенденции рынка;
4. адаптация стратегии найма;
5. оптимизация зарплат внутри компании.

Формирование итоговой презентации – здесь данные, полученные на предыдущем этапе, представляются в упорядоченном виде для упрощения восприятия аналитических данных. Здесь происходит верстка презентации, размещение смысловых акцентов. Этот этап нужен для подведения итогов аналитической работы.

Презентация перед руководством – на этом этапе отчет презентуют пред руководством, которое будет принимать решение по оптимизации бюджета на следующее полугодие.

Таким образом, для автоматизации процесса актуализации заработной платы сотрудников необходимо: разработать систему, которая автоматически собирает данные с hh.ru и формирует отчет по запросу пользователя. Тем самым сократив время всего процесса примерно на две трети.

## 1.3 Анализ конкурентов

Главным приемуществом нашего продукта по сравнению с конкурентами следует считать возможность формировать специфические метрики. Далее приводится список ресурсов, которые уже используются для анализа рынка. Также приводится список их приемуществ и недостатков.

1. Habr Карьера – электронный ресурс [3];
2. HeadHunter – электронный ресурс [4];
3. Rit.work – электронный ресурс [5];
4. Роснавык – электронный ресурс [6];
5. Зарплатомер (SuperJob) – электронный ресурс [7].

Система получения данных на Habr–карьера заключается в следующем. Пользователь, который хочет узнать ситуацию на рынке труда – должен указать свой уровень заработной платы и должность (профессию). Таким образом пользователь попадает в общий набор данных, который и предоставляется к анализу на этом портале.

То есть – ничто не мешает пользователю внести случайные данные, проверить это не выйдет, т.к. хабр предоставляет только итоговые данных, которые являются обезличенными.

Итоговая информация представляет собой набор квалификаций (Lead, Senior, Middle, Junior, Intern) – и соответствующее распределение зарплат. Также имеется возможность фильтрации по местоположению, специализации, профессиональным навыкам, компаниям, типу занятости и формату работы.

HeadHunter – самый популярный сайт для работодателей и соискателей в СНГ. Источником данных являются реальные заказчики рабочей силы. Каждую вакансию можно проверить на подлинность.

Итоговой информацией здесь является набор вакансий, которые можно свободно фильтровать по различным критериям.

Его недостаток лишь в том, что данные представлены как есть, то есть в слабо упорядоченном виде и наложить дополнительный фильтр на данные или собрать метрику (например медиану по выборке) – нельзя. Отчего нужно создавать свои инструменты анализа по данным, которые HeadHunter предоставляет.

Rit.work – это HeadHunter, но только для IT–вакансий. Здесь своя база данных, преимущество над HeadHunter`ом в том, что помимо тех же фильтров, что и на вышеупомянутой платформе, есть вывод метрики по отфильтрованным данным (перцентили, график – зарплата по месяцам, график – количество/зарплата).

Количество вакансий по сравнению с ХХ заметно меньше и обновляются они менее динамично, чем на самом популярном поисковике СНГ.

Роснавык обладает хорошей визуализаций данных и удобными отчетами. Однако нельзя «провалиться» в вакансию и узнать её источник. Ограниченное количество фильтров для одного запроса.

Также есть платные аналитические отчеты по согласованному списку профессиональных областей (до 4ех штук) по 50000 рублей за одну штуку.

Источники информации: Работа России, HeadHunter, SuperJob, Работа.ру. Проверки на дубли не ведется, отсюда можно получить погрешности при анализе данных.

SuperJob – удобные полезные отчеты по запрошенной профессии. Обезличенные данные. Высокая цена.

Результаты данного анализа можно представить в виде таблицы 1, где Парсер – название нашего сервиса.

Таблица 1 – Сравнительный анализ конкурентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сервис | Обезличенность | Стоимость | Метрики | Наполнение |
| Habr Карьера | 1 | 5 | 3 | 2 |
| HeadHunter | 5 | 5 | 1 | 5 |
| Rit.work | 5 | 5 | 3 | 2 |
| Роснавык | 1 | 3 | 5 | 3 |
| SuperJob | 1 | 1 | 4 | 5 |
| Парсер v1.0.0 | 5 | 4 | 5 | 5 |

# 2. Анализ средств разработки

## 2.1 Архитектура приложения

Программный продукт представляет собой два сервиса, связанных друг с другом технологией REST. Это означает, что каждый запрос от клиента к серверу должен содержать всю необходимую информацию, без использования данных о предыдущих запросах.

На рисунке 5 изображена схема двух сервисов, которые являют собой систему обработки вакансий с hh.ru и обеспечивают вывод информации в виде xlsx документов.

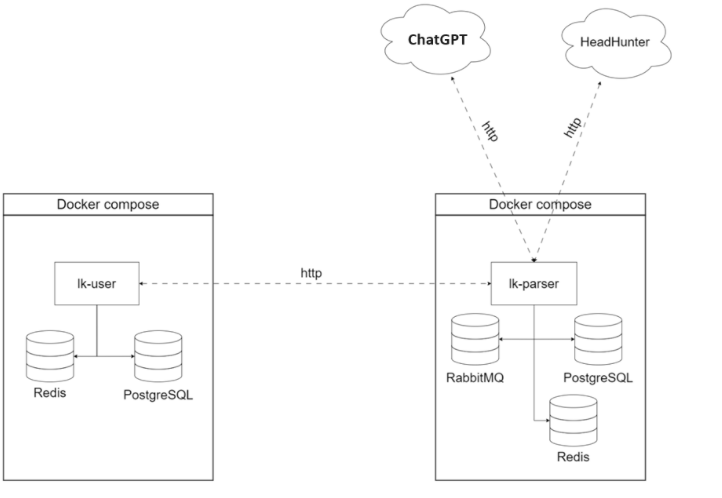


Рисунок 5 – Архитектура взаимодействия сервисов

В рамках данного продукта в этой работе описан личный кабинет пользователя (lk–user). Иными словами – верстка шаблонов, дизайн интерфейсов, добавление динамики при помощи средств JavaScript, внедрение JavaScript файлов в монолит при помощи фреймворка Webpack для соблюдения принципов чистого кода. В этом сервисе не представлено никакой сложной логики. Просто набор интерфейсов и база данных для хранения сопутствующих сущностей.

Второй сервис (сервис аналитики – lk–parser) – представляет собой парсер, генератор отчетов, калькулятор аналитической информации, прокси запросов к лингвистической модели и также хранилище данных для аналитики.

Взаимодействие между сервисами осуществляется посредством протокола HTTP [N], где необходимая информация передается в формате JSON, что соответствует принципам архитектуры REST. Каждый сервис оперирует собственным набором баз данных, обеспечивая тем самым высокую степень автономности. Это означает, что в случае неполадок или отказа одного из сервисов, другие могут продолжать свою работу без существенного влияния на общую функциональность системы.

Redis, представляющая собой NoSQL базу данных, служит для эффективного управления кэшем данных в распределенных средах. Она обеспечивает высокую производительность за счет хранения данных в оперативной памяти и использования сложных алгоритмов кэширования. Это позволяет уменьшить нагрузку на основные базы данных и сократить временные задержки при обращении к данным.

PostgreSQL, в свою очередь, является реляционной базой данных, предназначенной для долгосрочного и структурированного хранения данных. Ее гибкий и мощный SQL–движок обеспечивает эффективное хранение и обработку данных в различных сценариях, от малых приложений до крупных корпоративных систем.

RabbitMQ, являясь брокером сообщений, выполняет роль посредника между различными компонентами системы, обеспечивая надежную и отказоустойчивую передачу сообщений. Его механизмы хранения сообщений гарантируют сохранность данных даже в случае сбоев или отказов в работе сервисов, обеспечивая непрерывность работы системы и сохранение целостности данных. Каждый сервис находится в изолированном Docker контейнере, что позволяет обеспечить максимальную безопасность и устойчивость системы. Контейнеризация с помощью Docker обеспечивает изоляцию каждого сервиса, предотвращая взаимное воздействие между ними и минимизируя возможность уязвимостей. Кроме того, такой подход упрощает развертывание, масштабирование и управление сервисами, делая процесс более эффективным и прозрачным.

## 2.2 База данных

Для выбора базы данных необходимо провести сравнительный анализ между популярными движками. Для сравнения были выбраны: PostgreSQL, MariaDB, MySQL, Oracle Database, Microsoft SQL Server.

PostgreSQL – мощная, объектно-реляционная база данных с открытым исходным кодом, известная своей надежностью и производительностью. Она поддерживает многие современные функции, включая:

1. расширяемость (поддержка пользовательских типов данных, функций, операторов и индексов);
2. полная поддержка ACID транзакций;
3. продвинутая поддержка параллелизма;
4. развитые инструменты репликации и масштабирования;
5. нативная поддержка JSON и XML данных.

MariaDB – форк MySQL, также с открытым исходным кодом, популярный благодаря совместимости с MySQL и дополнительным функциям. Некоторые ключевые особенности:

1. поддержка ACID транзакций;
2. совместимость с MySQL;
3. улучшенные функции для производительности и безопасности;
4. поддержка различных механизмов хранения данных (например, Aria, XtraDB и др.);
5. расширенные возможности репликации и кластеризации.

MySQL – одна из самых популярных реляционных баз данных с открытым исходным кодом, часто используемая в веб-приложениях. Основные характеристики:

1. широкая совместимость и поддержка со стороны сообщества;
2. простота настройки и использования;
3. различные механизмы хранения данных (InnoDB, MyISAM и др.);
4. поддержка ACID транзакций (InnoDB);
5. поддержка репликации и кластеризации.

Oracle Database – коммерческая объектно-реляционная база данных, известная своими высокими показателями производительности и надежности. Основные характеристики:

1. расширенные возможности управления данными;
2. поддержка ACID транзакций;
3. развитые инструменты репликации и кластеризации;
4. высокая безопасность и поддержка масштабируемости;
5. богатый набор функций для аналитики и обработки данных.

Microsoft SQL Server – коммерческая реляционная база данных от Microsoft, часто используемая в корпоративных средах. Основные характеристики:

1. тесная интеграция с экосистемой Microsoft;
2. поддержка ACID транзакций;
3. развитые аналитические и BI функции;
4. высокая безопасность и масштабируемость;
5. инструменты для разработки и администрирования, такие как SQL Server Management Studio (SSMS).

Таблица N – сравнительный анализ движков баз данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Характеристика | PostgreSQL | MariaDB | MySQL | Oracle Database | Microsoft SQL Server |
| Лицензия | Некоммерческая | Некоммерческая | Некоммерческая | Коммерческая | Коммерческая |
| Тип БД | Объектно-реляционная | Реляционная | Реляционная | Объектно-реляционная | Реляционная |
| Расширяемость | Высокая | Умеренная | Ограниченная | Высокая | Высокая |
| JSON поддержка | Есть | Есть | Через костыль | Есть | Есть |
| Репликация | Асинхронная и синхронная репликация | Асинхронная и мастер-мастер репликация | Асинхронная и синхронная репликация | Асинхронная и синхронная репликация | Асинхронная и синхронная репликация |
| Масштабируемость | Высокая, поддержка шардинга | Высокая, поддержка Galera Cluster | Высокая, поддержка кластеризации | Высокая | Высокая |
| Документация | Обширная и детализированная | Хорошая, но иногда уступает MySQL | Обширная и доступная | Обширная, но требует лицензии | Обширная, но требует лицензии |
| Производительность | Высокая | Высокая | Высокая для веб-приложений | Очень высокая для сложных задач | Высокая для корпоративных приложений |

Тогда, на основе данных, приведенных раннее, PostgreSQL является отличным выбором по ряду причин:

1. производительность: PostgreSQL предлагает высокую производительность для сложных запросов, что может быть полезно даже в небольших проектах, если планируется работа с неструктурированными данными или сложными аналитическими запросами;
2. расширяемость: PostgreSQL предоставляет возможность добавлять пользовательские типы данных, функции и операторы, что позволяет легко адаптировать базу данных под специфические нужды вашего проекта;
3. поддержка JSON: Нативная поддержка JSON и JSONB делает PostgreSQL особенно привлекательным для проектов, которые активно используют API и взаимодействуют с данными в формате JSON;
4. сообщество и документация: PostgreSQL имеет активное сообщество и обширную документацию, что облегчает решение возникающих проблем и ускоряет процесс разработки;
5. надежность и безопасность: Полная поддержка ACID транзакций и расширенные функции безопасности гарантируют надежность и сохранность данных.

В отличие от других баз данных, таких как MariaDB и MySQL, PostgreSQL предлагает более продвинутые возможности работы с данными, параллелизма и расширяемости, что делает его оптимальным выбором для вашего проекта. Oracle Database и Microsoft SQL Server предоставляют высокую производительность и надежность, но их коммерческая лицензия и сложность могут быть излишни для небольшого проекта.

Таким образом, PostgreSQL выделяется среди аналогов как наиболее подходящий вариант для этого проекта, предлагая оптимальное сочетание производительности, расширяемости и надежности.

## 2.3 Язык программирования

Было принято решение разрабатывать сервис личных кабинетов связкой языков JavaScript и PHP. В частности, фреймворков jQuery, Symfony.

Выбор использования связки JavaScript (JQuery) + PHP Symfony с интеграцией Webpack и Bootstrap обусловлен несколькими ключевыми факторами, которые обеспечивают эффективное производство и достижение целей сервиса:

Хотя jQuery многими считается устаревшей технологией – она не теряет своих многочисленных плюсов, которые перечислены в официальной документации [8]:

1. простота использования – jQuery предлагает простой синтаксис, что делает работу с DOM и событиями веб-страницы более интуитивной;
2. кроссбраузерная совместимость обеспечивает однородное поведение на различных браузерах, что делает выполнение кода стабильным;
3. богатая библиотека функций предлагает широкий спектр функций и плагинов для анимации, AJAX запросов, управления CSS и других задач, упрощая разработку;
4. эффективность позволяет создавать компактный и эффективный код за счёт методов выбора элементов DOM и массовой манипуляции элементами;
5. активное сообщество и поддержка – jQuery имеет огромное сообщество разработчиков и обширную документацию, обеспечивая доступ к множеству ресурсов для обучения и развития.

Такой же участи удостоен PHP, как язык, которого принято считать устаревшим, однако нужно просто правильно его использовать, тогда его недостатки становятся его плюсами. Вот, что пишут про этот язык сами его создатели в документации [9]:

1. PHP легко изучить и использовать для создания веб-приложений;
2. поддержка различных баз данных и богатый функционал делают PHP мощным инструментом для разработки;
3. PHP успешно применяется в создании крупных и масштабируемых веб-приложений;
4. большое сообщество и ресурсы PHP обеспечивает доступ к поддержке и развитию.

Тогда стоит приступить к обсуждению взаимодействия связки этих технологий.

Производительность и нагрузка: Учитывая, что сервис не испытывает высокой нагрузки и выполняет относительно простые задачи, выбор легковесных технологий таких как JavaScript (JQuery) и PHP Symfony позволяет достичь необходимого уровня производительности без излишней сложности и затрат.

Экономичность: Использование открытых и бесплатных технологий, таких как JavaScript (JQuery) и PHP Symfony, а также фреймворка Bootstrap, помогает сократить расходы на лицензирование и разработку. Это особенно важно, учитывая дешевизну производства.

Удобство разработки и поддержки: Symfony предоставляет мощные инструменты для разработки веб–приложений, обеспечивая высокую производительность и удобство в работе с базой данных и другими аспектами приложения. Следуя официальной документации Webpack [10], интеграция с Webpack позволяет управлять зависимостями JavaScript и оптимизировать их загрузку, а Bootstrap, судя по документации [11], предоставляет готовые компоненты и стили для быстрой разработки пользовательского интерфейса.

Выбранный стек технологий обеспечивает достаточную гибкость и масштабируемость для возможного расширения функциональности и увеличения нагрузки в будущем

Таким образом, использование связки JavaScript (JQuery) + PHP Symfony с интеграцией Webpack и Bootstrap обеспечивает оптимальное сочетание экономичности, производительности, удобства разработки и гибкости, соответствуя потребностям и возможностям нашего сервиса.

Сервис «lk-parser» - является чисто backend приложением, для реализации которого была выбрана связка spring framework с java.

Так как проект может располагаться на сервере и должен быть доступен удаленно, было выбрано четыре языка программирования для сравнения: PHP (Symfony), JavaScript (Node), Go, Java (Spring), производительности языков программирования в веб-апп задачах основаны на сравнении из статьи «Производительность I/O бэкэнда: Node vs. PHP vs. Java vs. Go» [12].

Для визуализации данных, ниже приведена таблица 3 – таблица сравнения этих языков.

Таблица 3 – Сравнение языков программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | PHP | Java | Go | JavaScript |
| Производительность | 3 | 4 | 5 | 4 |
| Экосистема | 4 | 5 | 3 | 4 |
| Асинхронное программирование | 0 | 1 | 1 | 1 |
| Сложность изучения | 5 | 2 | 1 | 5 |
| Масштабируемость | 3 | 4 | 5 | 4 |
| Сообщество | 5 | 5 | 2 | 3 |
| Поддержка | 3 | 5 | 5 | 4 |
| Итог | 23 | 26 | 22 | 25 |

## 2.4 Выбор лингвистической модели

Важной частью автоматизации является нейросеть по вычленению слов по заданному контексту. Есть много различных решений этой конкретной задачи. Нужно выбрать ту, которая устроила бы заказчика и была легко интегрируема в наш сервис.

Есть два подхода к реализации работы нейросети в сервисе:

* облачное решение,
* компиляция кода нейросети.

Компилирование нейросети ведет к высокой ресурсной нагрузке. Более подробно это рассмотрено в одноименной статье [13], судя по ней – вычисления происходят на графическом ядре. Отсюда следуют высокие расходы на инфраструктуру проекта. (В сравнении со стоимостью облачного решения)

Помимо поддержки инфраструктуры, для скомпилированной нейросети нужно иметь большой датасет для тренировки этой самой модели.

количество примеров в датасете должно быть около 50тыс единиц для более–менее валидной работы этой самой нейросети. Чем больше – тем лучше. Обработка одной вакансии занимает 1–2 минуты. Таким образом сбор датасета займет около 1250 часов.

Также, натренированная нейросеть не учитывает свободный словесный запрос по типу: «если в твоей выборке будет вакансия на тимлида, то не включай эту вакансию в выборку, просто исключи её.»

Нейросеть, которая не является большой лингвистической моделью – не способна воспринимать такие просьбы пользователя.

Далее необходимо решить какую именно реализацию LLM (Large Language Model) выбрать. На данный момент существует множество генеративных моделей. В таблице N приведена классификация на основе статьи «ОБЗОР СОВРЕМЕННЫХ ГЕНЕРАТИВНЫХ НЕЙРОСЕТЕЙ: ОТЕЧЕСТВЕННАЯ И ЗАРУБЕЖНАЯ ПРАКТИКА» [N]

Таблица N – Классификация нейросетей по их функционалу

|  |  |
| --- | --- |
| Функционал нейросети | Наименование нейросети |
| Генерация текста | CharacterAI, Bing AI |
| Создание изображений | Dall-E, Kandinsky, Шедеврум |
| Монтаж видео | Runway Gen-2, Nvidia Eye Contact |
| Работа со звуком | Adobe Enhance, Voice.AI |
| Производительность | DeepL, Any Summary, Grammarly |
| Развлечения | RoomGPT, Outfits AI |
| Универсальные | ChatGPT, YandexGPT, GigaChat |

Для проекта подойдут только универсальные сети. Из них только две заслуживающие внимания (за счет доступности и цены): отечественное решение – YandexGPT и, разработанный компанией OpenAI, ChatGPT – Сан-Франциско, Калифорния, США.

В общем случае YandexGPT оказывается дешевле и проще в реализации, однако, на момент написания работы YandexGPT не умеет отдавать данные в формате JSON также хорошо, как это делает ChatGPT. Независимо от чистоты и тщательности запроса отечественная модель часто галлюцинирует. Пример можно наблюдать на рисунке [N]. В запросе явно указано – не выводить ничего кроме данных в формате JSON, однако YandexGPT выдает лишнюю информацию – «Здесь дата – 31 января 2023 года, время – 15:30». Такие галлюцинации на момент тестирования модели возникали довольно часто по сравнению с ChatGPT, который отдает нужный формат без корректировок запроса.

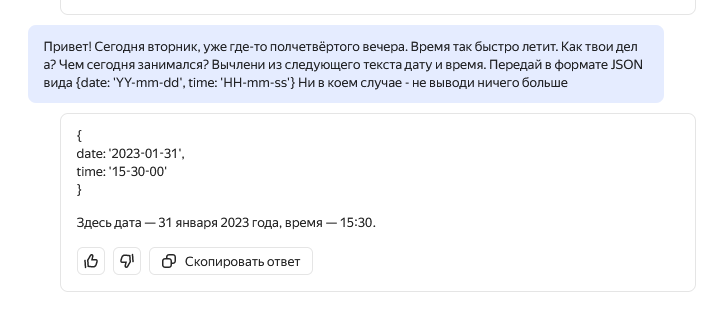


Рисунок [N] – пример галлюцинирования модели YandexGPT

Также использование ChatGPT обусловлено дешевизной разработки, то есть это уже готовое решение, которое свободно интегрируется посредством открытого API, которую OpenAI предоставляет свободно, судя по документации [14].

Т.к. прямые запросы в ChatGPT из России невозможны – нужно использовать proxy для запросов. Таким образом выбор пал на neuroapi, где стоимость запроса в 500 токенов (примерно столько занимает запрос одной вакансии) равна 0,02 рубля, исходя из их прейскуранта [15]. То есть при среднем объеме отчета в 600–1000 вакансий (запросов) и 120 должностях. Получаем расход, соответствующий диапазону

,

,

Тогда стоимость за одну реализацию X – процесса по актуализации заработной платы сотрудников «Artsofte» составляет от 1440 рублей до 2400 рублей.

Сервер с видеокартой для реализации LLM нейросети (или же упрощенной NLP – Natural Language Processing) стоит в месяц минимум 7500 рублей (самый дешевый тариф на hostkey.ru). Есть возможность брать сервера посуточно (около 400 рублей в сутки, что примерно равно использованию прокси), но тогда все запросы нужно будет формировать заранее и исключается возможность «экстренного парсинга», также это накладывает лишние расходы на тестирование и разработку продукта.

# 3. Техническое задание

Техническое задание представлено в приложении Б.1 и далее. Ключевая часть реализации, которая передана программисту серверной части сервиса «lk-parser» представлена в приложении В.5 и В.6. В этих главах описаны ключевые требования к разработке приложения.

Также большую часть технического задания занимают интерфейсы, которые были разработаны на стороне заказчика и доработаны по просьбе аналитики.

Сам скелет технического задания был основан на стандарте ГОСТ 19.201-78 [N]. Далее будут кратко описаны главы технического задания и их назначение:

1. введение. Раздел "Введение" в техническом задании необходим для установления контекста и создания общей картины проекта. Он должен предоставить читателю понимание целей и значимости проекта. Введение описывает общую проблему, которую проект решает, и кратко излагает его основные цели.
2. основания для разработки. Этот раздел фокусируется на причинах и предпосылках, побудивших к началу проекта. Здесь изложено - какие именно проблемы или потребности вызывают необходимость разработки, какие требования не удовлетворяются текущими решениями, и каковы ожидаемые выгоды от реализации проекта. Этот раздел включает анализ существующих условий, статистические данные, результаты исследований или опросов, которые подкрепляют необходимость разработки. Таким образом, этот раздел служит аргументированной базой для начала работы над проектом.
3. назначение разработки. В этом разделе четко определяются цели и задачи, которые должен решить проект. Он описывает, какие конкретные результаты ожидаются от разработки, и как они будут способствовать решению обозначенных проблем или удовлетворению потребностей. В этом разделе указаны основные функции, которые будет выполнять разрабатываемое решение, его ключевые характеристики и параметры. Четкое определение назначения разработки помогает установить критерии успеха проекта и обеспечивает общий фокус для всех участников. Это также служит ориентиром для разработки технических спецификаций, планирования этапов работы и оценки достигнутых результатов.
4. требования к программе. Раздел "Требования к программе" определяет все аспекты разработки и функционирования ПО, включая задачи, обрабатываемые данные и технические аспекты. Подраздел "Требования к составу выполняемых функций" описывает основные и дополнительные функции программы, а "Требования к источникам данных" определяет типы данных, их формат и источники. "Требования к реализации" охватывают используемые технологии, архитектуру и стандарты, включая совместимость и безопасность. "Требования к организации входных и выходных данных" описывают форматы, способы ввода и вывода данных, а "Требования к временным характеристикам" устанавливают время отклика, выполнение операций и масштабируемость. Этот раздел обеспечивает прозрачность и согласованность ожиданий, служит основой для разработки, тестирования и внедрения программы.
5. требования к надёжности. Здесь определены критерии для обеспечения стабильного функционирования программы, включая методы минимизации отказов и устойчивости работы. Подраздел "Требования к обеспечению надежного функционирования программы" описывает меры, такие как резервирование, отказоустойчивые архитектуры и процедуры автоматического восстановления. "Время восстановления после отказа" определяет максимальное время для восстановления работы после сбоя и методы восстановления. "Отказы из-за некорректных действий оператора" рассматривает механизмы для минимизации ошибок пользователей, такие как защита от ошибок и понятные инструкции. Этот раздел помогает заказчику эксплуатировать систему, обеспечивая бесперебойную работу и минимизацию рисков, связанных с человеческим фактором.
6. условия эксплуатации. Раздел "Условия эксплуатации" описывает требования для эффективного использования и обслуживания программы. Подраздел "Требования к видам обслуживания" определяет виды и частоту необходимого обслуживания для поддержания работоспособности системы, включая профилактические проверки и обновления. Подраздел "Требования к численности и квалификации персонала" описывает необходимое количество и уровень квалификации сотрудников, которые будут заниматься эксплуатацией и обслуживанием программы, чтобы обеспечить её бесперебойное функционирование.
7. требования к составу и параметрам технических средств. Раздел определяет необходимые аппаратные компоненты и их характеристики для обеспечения оптимальной работы программы. Этот раздел включает спецификации для серверов, рабочих станций, сетевого оборудования и других технических средств, а также требования к их производительности, надёжности и совместимости с программным обеспечением. Чёткое определение этих параметров позволяет обеспечить соответствие технических средств требованиям программы, что гарантирует её стабильную и эффективную работу.
8. требования к программной документации. Раздел определяет состав и содержание документации, необходимой для разработки и эксплуатации программы. Предварительный состав программной документации включает: техническое задание, программу и методику испытаний, описание процесса выпуска токенов. Эта документация обеспечивает разработчиков, тестировщиков и пользователей подробной информацией о программе, её функциональных возможностях и процессах, что способствует эффективному использованию и обслуживанию системы.
9. технико-экономические показатели. Раздел включает в себя анализ экономической целесообразности проекта, оценку затрат на разработку и внедрение программного продукта, а также прогнозирование экономических выгод от его использования.
10. стадии и этапы разработки. Раздел предоставляет план работы, определяя последовательность шагов, необходимых для создания и внедрения программного продукта. Определение стадий и этапов разработки позволяет контролировать прогресс, выявлять возможные риски и препятствия, а также распределять ресурсы эффективно. Кроме того, этот раздел служит основой для планирования и управления проектом, обеспечивая его успешное завершение в соответствии с поставленными целями и требованиями.
11. порядок контроля и приемки. Раздел определяет процедуры и критерии, по которым будет осуществляться проверка готовности программного продукта к внедрению и его последующему приему. Этот раздел является ключевым для обеспечения качества и соответствия разработанного продукта поставленным требованиям. Здесь описываются этапы тестирования, включая функциональное, интеграционное и приемочное тестирование, а также критерии успешного прохождения каждого этапа. Также здесь определяются ответственные лица и сроки проведения контрольных мероприятий.

# 4. Проектирование

## 4.1 Обновленный процесс сбора наглядной аналитики

Существующий процесс представлен на рисунке 3. Его подробное описание выглядит так: собираем список должностей для анализа. Обычно этих ролей около 1, в связи с этим работа по обработке разделяется на несколько человек. Каждый HR берет из «стопки должностей» одну и формирует по ней xlsx файл: сначала по названию должности подается запрос в hh.ru, чтобы найти актуальные вакансии по этому названию. Каждая вакансия соответствует строке xlsx файла. Сама строка формируется путем сбора следующей информации из описания вакансии:

1. извлечение задач и функционала;
2. извлечение информации о дополнительных условиях труда;
3. извлечение результирующего уровня заработной платы,
4. извлечение дополнительных данных для верстки отчета.

После того, как по каждой вакансии сформирована строка – xlsx файл считается законченным. И берутся следующие должности до тех пор, пока они не закончатся.

Тогда определим слабые стороны старого процесса:

1. над обработкой ролей работает несколько человек;
2. каждая отдельная вакансия обрабатывается в несколько сложных этапов;
3. основное время уходит на рутинные действия, которые легко автоматизировать.

Следовательно, в новом процессе эти недостатки должны быть исправлены. Тогда новый процесс должен выглядеть как на рисунке 12:

A diagram of a process

Description automatically generated

Рисунок 12 – обновленный процесс получения отчета

Теперь процесс выглядит так: получаем выгрузку должностей из битрикса, выполняем запрос на формирования отчета по каждой должности. Далее запросы попадают в сервис формирования отчетов. (Там, по определенному запросу в LLM, автоматизируются ручные действия по вычленению данных из текста.) Ожидаем выполнения всех запросов. Данные собраны и представлены в xlsx формате, содержание отчета представлено в приложении В.1.

## 4.2 User–flow

Для демонстрации работы с продуктом было принято решение воспользоваться руководством по созданию user–flow [17]. User–flow (поток пользователя) представляет собой последовательность действий, которые пользователь выполняет при взаимодействии с продуктом или сервисом (рисунок 13). Это концептуальная модель, которая описывает путь пользователя от начальной точки (например, посещение веб–сайта или запуск приложения) до достижения конечной цели (например, совершение покупки, регистрация аккаунта и т. д.).

Изображение выглядит как диаграмма, снимок экрана, линия, План

Автоматически созданное описание

Рисунок 13 – user flow получения отчета

Принцип работы с нашим решением довольно прост и не требует дополнительного обучения персонала: чтобы получить отчет необходимо войти в личный кабинет, используя логин и пароль, далее нужно зайти во вкладку Request (рисунок 14) и заполнить нужные для запроса поля.

A screenshot of a computer

Description automatically generated

Рисунок 14 – форма запроса

Запрос описан в таблице Г.1 Стоит отметить, что в каждом поле с мультиселектом реализован поиск по подстроке и добавлены кнопки для выбора всех вариантов. Также из–за большого списка индустрий с hh.ru было принято решения добавить кнопку для их сокрытия чтобы не перегружать интерфейс.

После того, как запрос был отправлен на сервер, можно зайти во вкладку Documents (рисунок 15) и увидеть статус документа:

1. START\_PARSING – парсинг начат;
2. COMPLETE – парсинг окончен;
3. GENERATE\_DOCUMENT – генерация документа;
4. COMPLETE\_WITH\_DOCUMENT – генерация документа окончена;
5. PARSING\_ERROR – ошибка при выполнении.

A screenshot of a computer

Description automatically generated

Рисунок 15 – страница со списком отчетов

Также при нажатии на хэш можно посмотреть запрос более детально (рисунок 16), что позволяет получить дополнительную информацию о конкретном запросе или его результате. Это особенно полезно для анализа данных и выявления паттернов или аномалий в информации.

A screenshot of a computer

Description automatically generated

Рисунок 16 – страница с содеранием запроса

Таким образом user–flow укладывается в три интуитивных интерфейса и не требует привлечения дополнительных ресурсов на обучение персонала. Главным преимуществом этого порядка действий является то, что не нужно дожидаться выполнения одного запроса, чтобы начинать формировать следующий.

## 4.3 Проектирование базы данных сервиса «lk-user»

Для хранения полезных данных была предложена следующая схема базы данных сервиса личных кабинетов (рисунок А.1).

База данных сервиса личного кабинета состоит из 6 сущностей:

1. пользователь – чтобы обеспечивать ограниченный доступ к ресурсу;
2. миграции – для удобства развертывания приложения и актуализации базы данных при разработке;
3. hh–регион – регионы хранятся в базе данных чтобы не запрашивать их с hh при каждом формировании аналитического запроса;
4. hh–индустрия – аналогично региону;
5. hh–подиндустрия – аналогично региону, связана с hh–индустрией первичным ключом;
6. аналитический запрос – аналитический запрос хранится на стороне личного кабинета чтобы можно было переотправлять запросы при неудачном выполнении.

На рисунке Б.1 представлена база данных сервиса lk–parser, ниже представлено её описание.

База данных сервиса аналитики опирается на 5 сущностей (плюс 3 вспомогательных).

Сущность вакансии (vacancy) – аналитическая единица отчета, именно эта сущность обогащается данными при парсинге и дополнительной обработке.

Также в отдельную таблицу (order\_excel\_file\_param) записываются параметры формируемых файлов для более удобного обращения к ним при скачивании.

Formula\_setting и calculation\_results служат для сохранения применяемых к вакансии формул при обработке.

Также существует таблица, дублирующая таблицу из сервиса личного кабинета – parser\_order. Это сделано для репликации данных, чтобы в случае чего не было потерь важной информации.

Отдельно достойна описания ключевая сущность – сущность аналитического запроса.

Сущность аналитического запроса (analytics\_request) служит для хранения запросов пользователей. В таблице Д.1 приведены поля этой сущности с пояснениями:

## 4.4 Проектирование клиент-серверного взаимодействия сервиса

Межсервисное взаимодействие частично представлено на рисунке 5. В этом пункте описано взаимодействие внутри сервиса «lk-user» и часть межсервисного взаимодействия.

На рисунке 17 можно заметить, что lk-user frontend и lk-user backend общаются путем «внутреннего API». Это означает, что обмен данными происходит путем шаблонизации HTML документов при помощи библиотеки twig. То есть без HTTP запросов к бекенду.

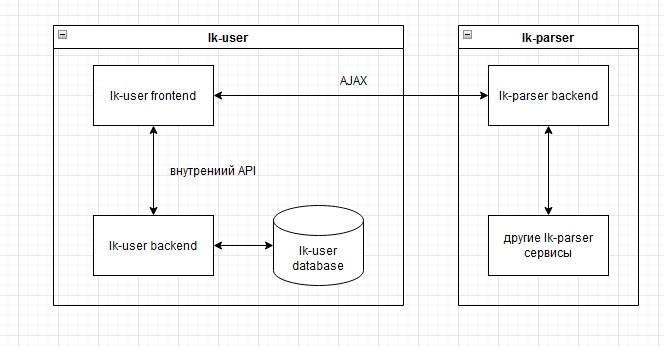


Рисунок 17 – Внутреннее и внешнее взаимодействие «lk-user» и «lk-parser»

А внешнее взаимодействие как раз происходит путем HTTP запросов при помощи AJAX запросов, которые являются частью библиотеки jQuery.

Таким образом на фронте не нужно реализовывать методы авторизации, не нужно отдельно обрабатывать валидацию форм. Т.к. все это реализовано на стороне бекенда Symfony Bundle, и попадают на клиентскую часть через twig шаблоны. JavaScript скрипты нужны для обеспечения динамики форм. И внешних запросов.

## 4.5 Проектирование архитектуры «lk-user»

Под архитектурой здесь понимается компоновка файлов внутри проекта. Скелет всего проекта представлен на рисунке 18

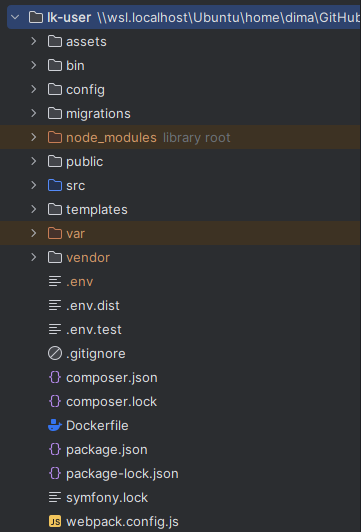


Рисунок 18 – Основной скелет проекта

Функциональность такого расположения файлов – следующая:

1. assets – тут располагается клиентская часть приложения;
2. src – тут располагается серверная часть приложения;
3. public – тут располагается статика приложения;
4. var, vendor, node\_modules – служат для хранения библиотек;
5. .env – файлы окружения;
6. composer.json, package.json – соответственно конфигурация php, javascript;

Далее прилагается скелет серверной части приложения. На рисунке 19 предоставляется содержание папки src. Само содержание подчиняется стандартному скелету Symfony – проекта.

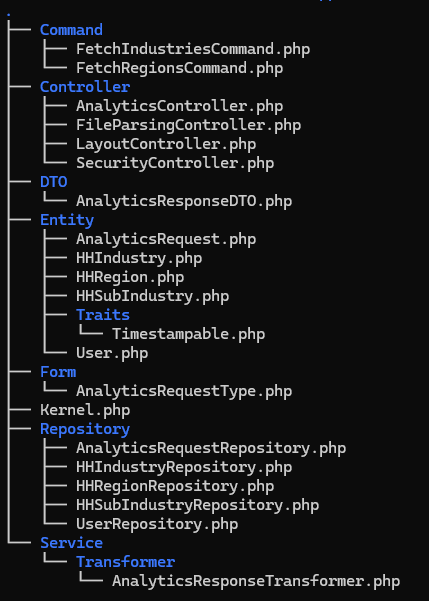


Рисунок 19 – Содержание папки src

Функциональность такого расположения файлов – следующая:

1. command – здесь лежат команды для актуализации индустрий и регионов с hh.ru;
2. controller – здесь располагаются контроллеры для различных действий пользователя;
3. DTO – здесь лежат объекты для унификации передачи данных;
4. entity – эти файлы являются моделями для базы данных;
5. form – формы запросов;
6. repository – через репозитории происходит доставка данных из базы данных;
7. service – различные дополнительные сервисы.

Также в папке assets на рисунке 20 – располагаются JavaScript скрипты: из них (и шаблонов из папки templates) и состоит клиентская часть приложения.

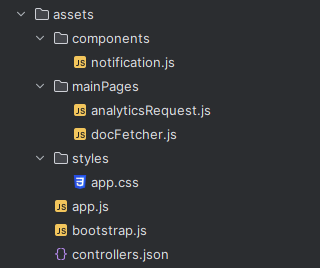


Рисунок 20 – папка assets

Webpack предоставляет удобную маршрутизацию файлов внутри проекта, таким образом можно настроить удобный доступ до файлов внутри шаблонов. Название папок говорит само за себя:

1. components – здесь лежат компоненты, которые следует переиспользовать в разных частях скриптов;
2. mainPaiges – здесь лежат скрипты для главных страниц приложения;
3. styles – здесь располагаются стили приложения;
4. app.js – точка входа в приложение;
5. bootstrap.js – интеграция UI/UX библиотеки bootstrap;
6. controllers.json – конфигурация маршрутизации контроллеров.

# 5. Реализация

## 5.1 Реализация клиентской части приложения

Специфика реализации клиентской части приложения в этом сервисе является то, что клиентская часть не является отдельным приложением со своим сервером. А просто сбор статики рядом с бекендом на Symfony 6. Это сделано для удешевления разработки и простоты поддержки. Чтобы запустить клиентскую часть «рядом» были предприняты следующие шаги:

Написан Dockerfile, в котором отдельно учтено условие, когда на сервере отсутствует nodejs (node\_modules). Таким образом достаточно запустить Dockerfile и не разворачивать модули отдельно.

FROM node:14-alpine

WORKDIR /lkuser

COPY package.json ./

COPY package-lock.json ./

RUN npm install

//условие, упрощающее разворачивание модулей

CMD [ -d "node\_modules" ] && npm run dev-server || npm ci && npm run dev-server

Теперь чтобы Javascript собирался как статические файлы рядом с php, который тоже собирается как статические файлы, необходимо использовать сборщик модулей. В этом проекте был выбран Webpack — это бесплатный сборщик модулей с открытым исходным кодом для JavaScript. Он создан в первую очередь для JavaScript, но может преобразовывать внешние ресурсы, такие как HTML, CSS и изображения, если включены соответствующие загрузчики. Webpack принимает модули с зависимостями и генерирует статические ресурсы, представляющие эти модули.

Настройка Webpack происходит автоматически при помощи библиотеки «@symfony/webpack-encore». То есть после установки командой composer require symfony/webpack-encore-bundle – файл конфигурации не менялся.

Важными частями клиентской части являются: обновление статуса документов, страница формирования аналитического запроса.

Обновление статуса документов реализована через long polling – используется в веб-приложениях реального времени для достижения практически мгновенной связи между клиентом и веб-сервером. Это особенно полезно в приложениях чата и обмена сообщениями, где обновления в реальном времени имеют решающее значение.

Запрос по документам на странице отправляется в «lk-parser» и записи на странице обновляются без обновления страницы. Метод, который отправляется каждые 5 секунд отображен в приложении Е.1. В самом методе вставляется HTML код по ключевым тегам на странице в зависимости от статуса документа.

Форма формирования запроса имеет multiselect поля, которые должны быть реализованы через checkbox, что означает, что из множества пунктов, должно быть выбрано несколько одновременно. Для реализации этой функциональности была выбрана библиотека select2. Также эта библиотека поддерживает текстовый поиск, что сильно улучшает пользовательский опыт. Конфигурация происходит непосредственно при вызове метода select2.

$('#searchModifier').select2({

width: '100%',

placeholder: 'Выберите модификатор поиска',

allowClear: true,

closeOnSelect: false,

theme: "bootstrap-5"

});

Ошибки при отправке формы обрабатываются на стороне Symfony. И выводится через JavaScript. Ошибки автоматически обрабатываются в зависимости от типа, дополнительно наложено ограничение на пустое поле поиска. Пример валидации на стороне сервера представлен ниже.

public function buildForm(FormBuilderInterface $builder, array $options): void

{

$builder

->add('searchField', TextType::class, [

'constraints' => [

new NotBlank(['message' => 'Поле поиска не должно быть пустым']),

],

])

->add('qualificationLevel', TextType::class)

->add('region', CollectionType::class)

->add('vmi', CheckboxType::class)

->add('searchModifier', CollectionType::class)

## 5.2 Реализация серверной части приложения

На рисунке 19 изображён скелет проекта. Если идти сверху вниз по файлам проекта, то функциональность они будут нести следующую:

Command – в проекте реализованы две команды, которые шлют запросы в hh.ru и актуализируют (сохраняют в базу данных) список регионов и список индустрий.

Controller – контроллеры существуют чтобы обозначит маршрутизацию в приложении. В контроллерах реализованы все url – пути, которые присутствуют в приложении.

Entity – служат для реализации представления моделей в базе данных. В «lk-user» используется DoctrineORM, таким образом инициализация сущности выглядит как на примере ниже.

#[ORM\Entity(repositoryClass: HHIndustryRepository::class)]

class HHIndustry

{

#[ORM\Id]

#[ORM\GeneratedValue]

#[ORM\Column]

private ?int $id = null;

#[ORM\Column(length: 255, nullable: true)]

private ?string $hhId = null;

#[ORM\Column(length: 255, nullable: true)]

private ?string $name = null;

Благодаря подходу с описанием сущностей при помощи DoctrineORM – команда разработки имеет следующие преимущества:

1. автоматизация процесса миграций: Doctrine автоматически генерирует скрипты миграций на основе изменений в сущностях. Это снижает вероятность ошибок и экономит время, так как разработчику не нужно вручную писать SQL-запросы для обновления схемы базы данных.
2. историчность изменений: Все изменения в структуре базы данных сохраняются в виде отдельных миграций. Это позволяет откатываться к предыдущим версиям схемы базы данных, если это необходимо, и упрощает отслеживание изменений.
3. интеграция с системой контроля версий: Миграции хранятся в системе контроля версий (например, Git), что позволяет разработчикам работать в команде и синхронизировать изменения в базе данных вместе с изменениями в коде.
4. управление зависимостями: Doctrine ORM учитывает зависимости между сущностями и автоматически генерирует миграции в правильном порядке, что предотвращает ошибки в базе данных при применении изменений.

Помимо упомянутых преимуществ, на основе DoctrineORM реализован паттерн repository – суть которого в том, что доступ к сущностям базы данных унифицирован. А подключая ORM запросы вместо чистого SQL – серверная часть становится защищенной от атаки вида PHP CODE INJECTION, упомянутой в статье «ЗАЩИТА ВЕБ-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТИ PHP CODE INJECTION» [17].

# ЗАКЛЮЧЕНИЕ

Разработка и внедрение системы HRParser позволило компании Artsofte решить проблему актуализации заработной платы сотрудников, оптимизировать расходы и повысить эффективность работы HR отдела. Автоматизация данного процесса стала ключевым фактором для улучшения управления персоналом и обеспечения конкурентоспособности компании на рынке труда

Анализ предметной области и выявление проблемы позволили сфокусироваться на разработке инновационного решения, которое значительно упростило и ускорило процесс актуализации заработной платы. Теперь HR отдел имеет возможность чаще проводить анализ рынка труда и обновлять данные о заработной плате сотрудников, что способствует более точному прогнозированию расходов компании и повышению мотивации персонала.

При разработке системы были выполнены следующие задачи:

1. изучение процессов организации встречи в компании Artsofte;
2. выявление и описание требований к системе у заказчика;
3. описание функциональных требований к системе;
4. описание сценариев использования;
5. подготовка и передача технического задания в разработку.

Внедрение HRParser открывает новые перспективы для компании Artsofte, позволяя ей оперативно реагировать на изменения на рынке труда, улучшать условия работы сотрудников и обеспечивать устойчивое развитие в условиях динамичного бизнес–окружения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. TalentTech: офиц. сайт. – Москва. – URL: https://main.talenttech.ru/ (дата обращения: 05.04.24)
  2. Хабр (М.Тех – М.Видео–Эльдорадо): офиц. сайт. – Москва. – URL https://habr.com/ru/companies/mvideo/articles/732966/ (дата обращения 05.04.24).
  3. Информация по конкурентному сервису Хабр–карьера: офиц. сайт. – Москва. – URL: https://career.habr.com (дата обращения 05.04.24).
  4. Информация по конкурентному сервису hh.ru: офиц. сайт. – Москва. – URL: https://hh.ru (дата обращения 05.04.24).
  5. Информация по конкурентному сервису Rit.work: офиц. сайт. – Москва. – URL: https://rit.work (дата обращения 05.04.24).
  6. Информация по конкурентному сервису Роснавык: офиц. сайт. – Москва. – URL: https://lk.rosnavyk.ru (дата обращения 05.04.24).
  7. Информация по конкурентному сервису SuperJob: офиц. сайт. – Москва. – URL: https://www.zarplatomer.ru (дата обращения 05.04.24).
  8. jQuery: офиц. сайт. – Сан-Франциско. – URL: https://jquery.com/ (дата обращения 05.04.24).
  9. PHP: офиц. сайт. – Сан-Франциско. – URL: https://www.php.net/ (дата обращения 05.04.24).
  10. Webpack: офиц. сайт. – Сан-Франциско. – URL: https://webpack.js.org/ (дата обращения 05.04.24).
  11. Build fast, responsive sites with Bootstrap: офиц. сайт. – Кембридж. – URL: https://getbootstrap.com/ (дата обращения 05.04.24).
  12. Производительность I/O бэкэнда Node vs. PHP vs. Java vs. Go : сайт - URL: https://habr.com/ru/companies/vk/articles/329258/ Дата обращения 05.04.2024).
  13. Локальные нейросети (генерация картинок, локальный chatGPT): офиц. сайт. – Москва. – URL: https://habr.com/ru/articles/727834/ (дата обращения 05.04.24).
  14. ChatGPT openAPI: офиц. сайт. – Сан-Франциско. – URL: https://chat.openai.com/ (дата обращения 05.04.24).
  15. HeadHunterAPI: офиц. сайт. – Москва. – URL: https://github.com/hhru/api/blob/master/docs/authorization\_for\_application.md/ (дата обращения 05.04.24).
  16. User flow: как создаются популярные приложения и сайты: офиц. сайт. – Москва. – URL: https://habr.com/ru/articles/496760/ (дата обращения 05.04.24).
  17. Беликов Георгий Витальевич, Крылов Илья Дмитриевич, Селищев Валерий Анатольевич ЗАЩИТА ВЕБ-ПРИЛОЖЕНИЙ ОТ УЯЗВИМОСТИ PHP CODE INJECTION // Известия ТулГУ. Технические науки. 2022. №5. URL: https://cyberleninka.ru/article/n/zaschita-veb-prilozheniy-ot-uyazvimosti-php-code-injection (дата обращения: 05.04.2024).
  18. И.О. Малышев, А.А. Смирнов ОБЗОР СОВРЕМЕННЫХ ГЕНЕРАТИВНЫХ НЕЙРОСЕТЕЙ: ОТЕЧЕСТВЕННАЯ И ЗАРУБЕЖНАЯ ПРАКТИКА // Международный журнал гуманитарных и естественных наук. 2024. №1-2 (88). URL: https://cyberleninka.ru/article/n/obzor-sovremennyh-generativnyh-neyrosetey-otechestvennaya-i-zarubezhnaya-praktika (дата обращения: 05.04.2024).
  19. Введение в протоколы HTTP и HTTPS [Электронный ресурс]: https://docs.microsoft.com/ru-ru/azure/rtos/netx-duo/netx-duo-web-http/chapter1(дата обращения: 05.04.2024).
  20. ГОСТ 19.201-78 [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200007648 (Дата обращения 05.04.2024).

# ПРИЛОЖЕНИЕ А (обязательное) Схема базы данных сервиса ЛК

A screenshot of a computer

Description automatically generated

Рисунок А.1 – База данных сервиса личных кабинетов

# ПРИЛОЖЕНИЕ Б (обязательное) Схема базы данных сервиса аналитики

A screenshot of a computer

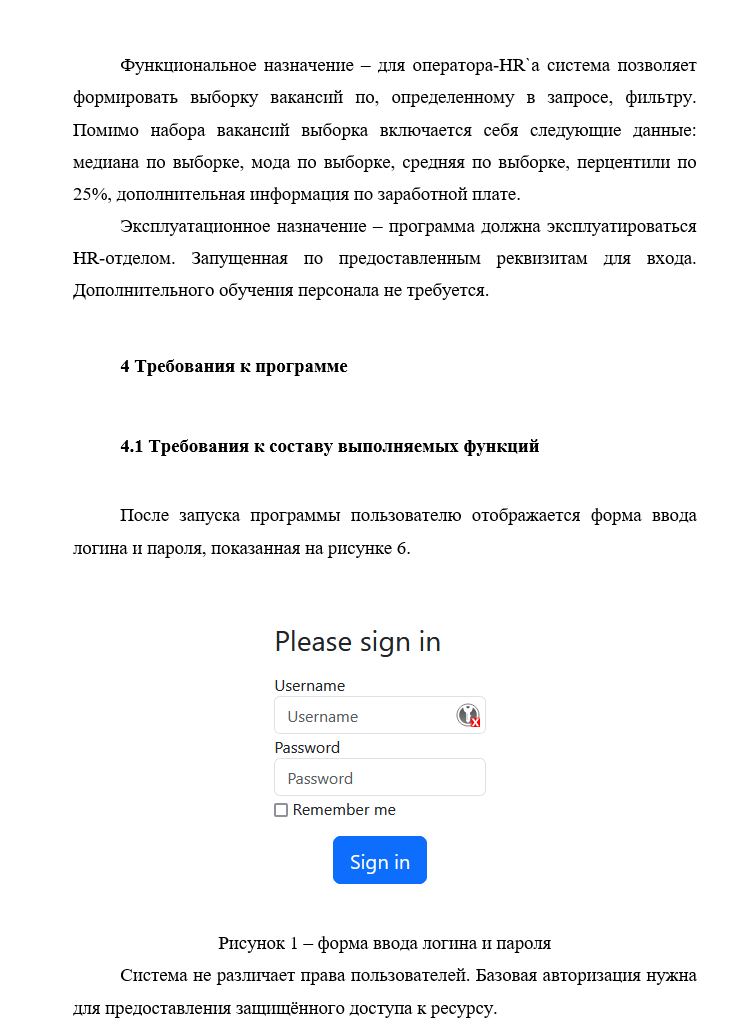
Description automatically generated

Рисунок Б.1 – База данных сервиса аналитики

# ПРИЛОЖЕНИЕ В (обязательное) Техническое задание проекта



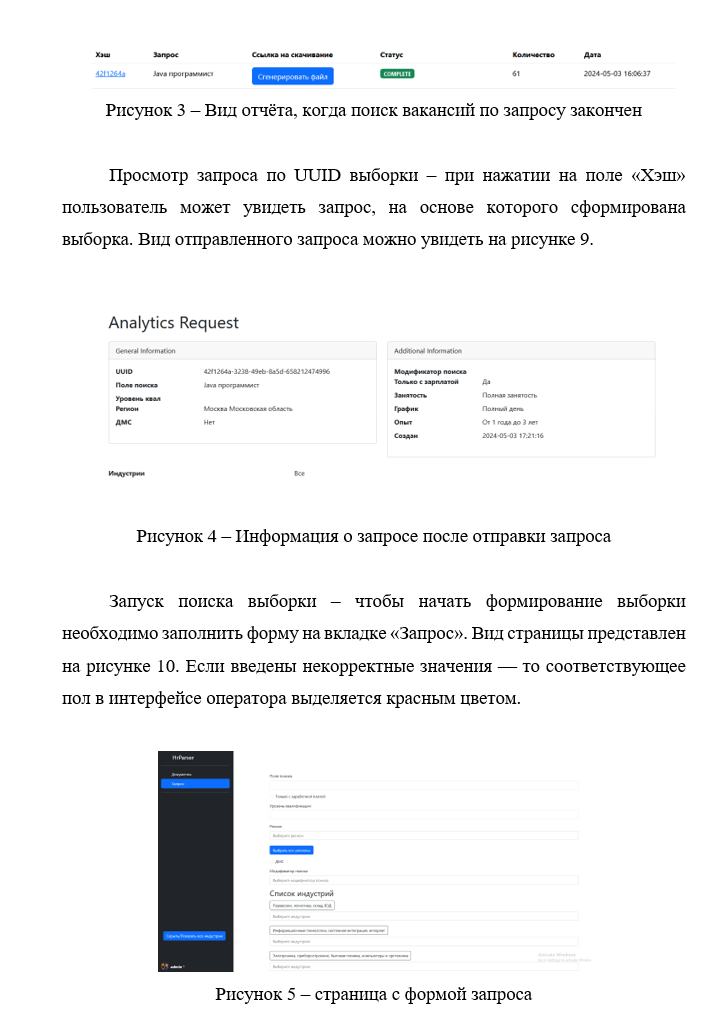
**Продолжение ПРИЛОЖЕНИЯ В**

****

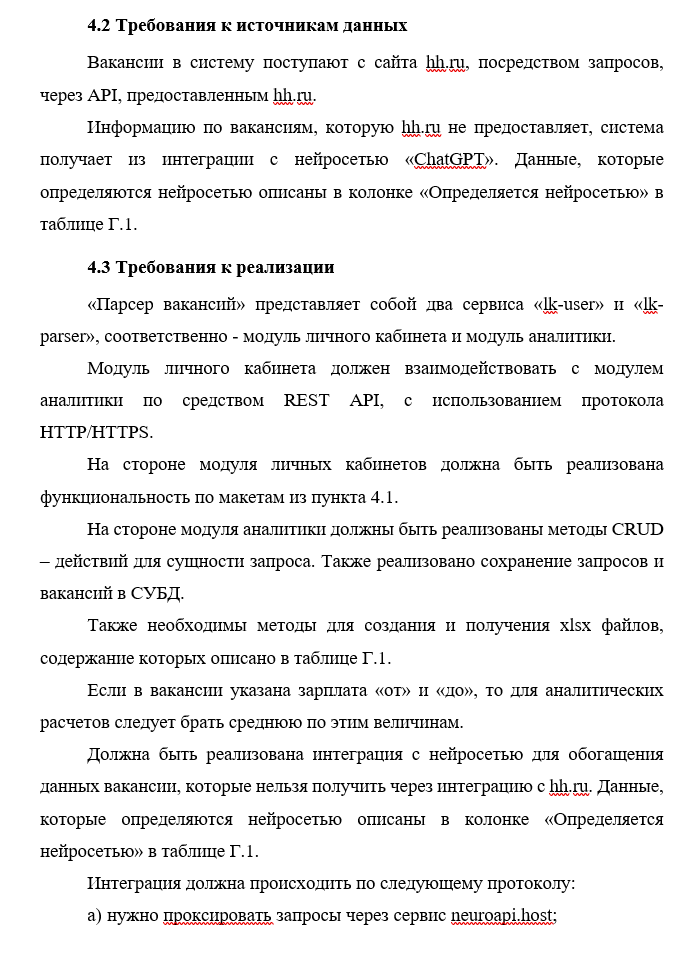
**Продолжение ПРИЛОЖЕНИЯ В**



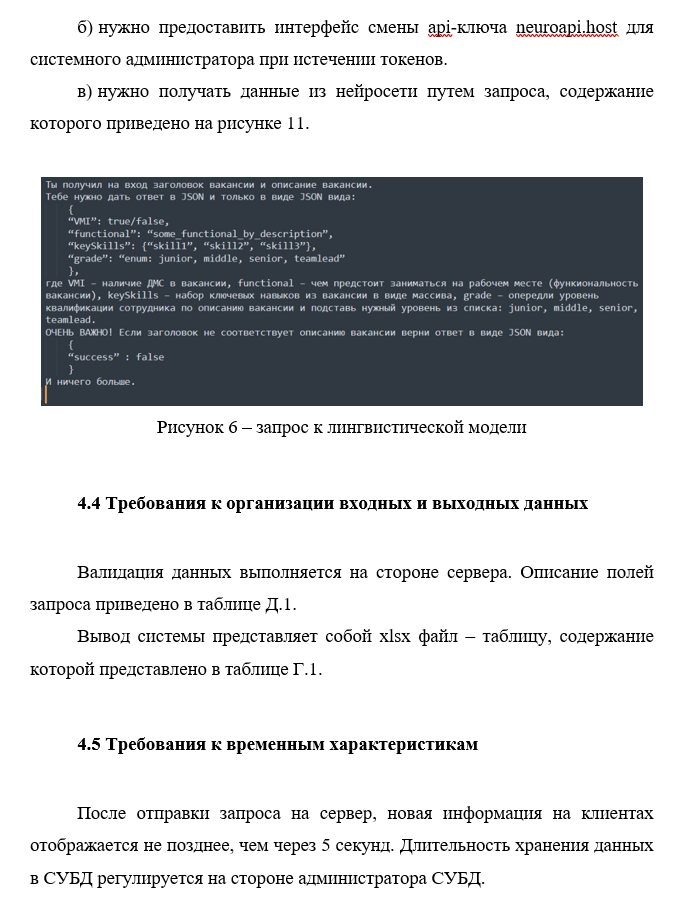
**Продолжение ПРИЛОЖЕНИЯ В**



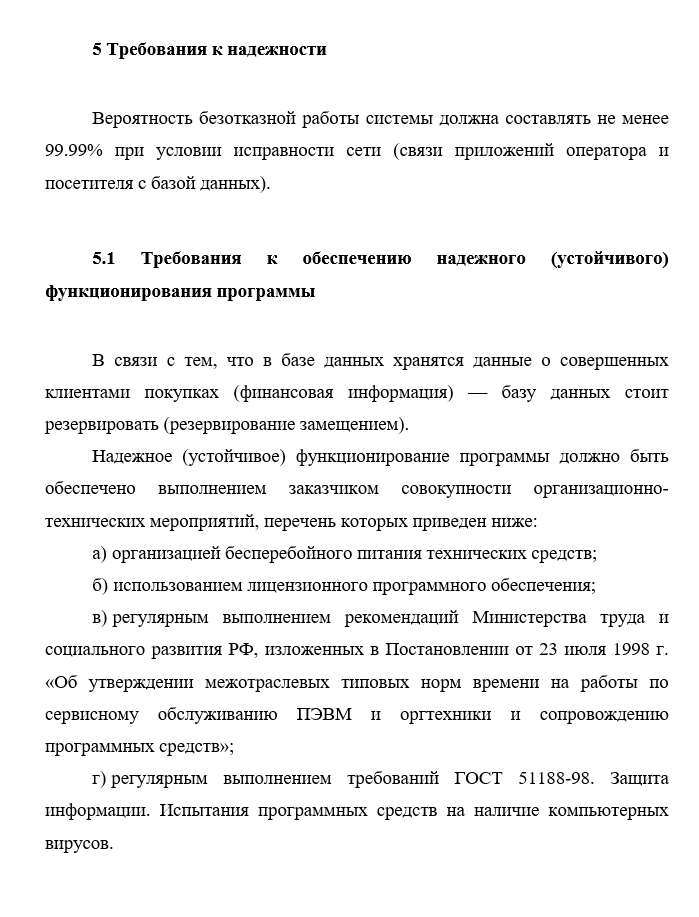
**Продолжение ПРИЛОЖЕНИЯ В**



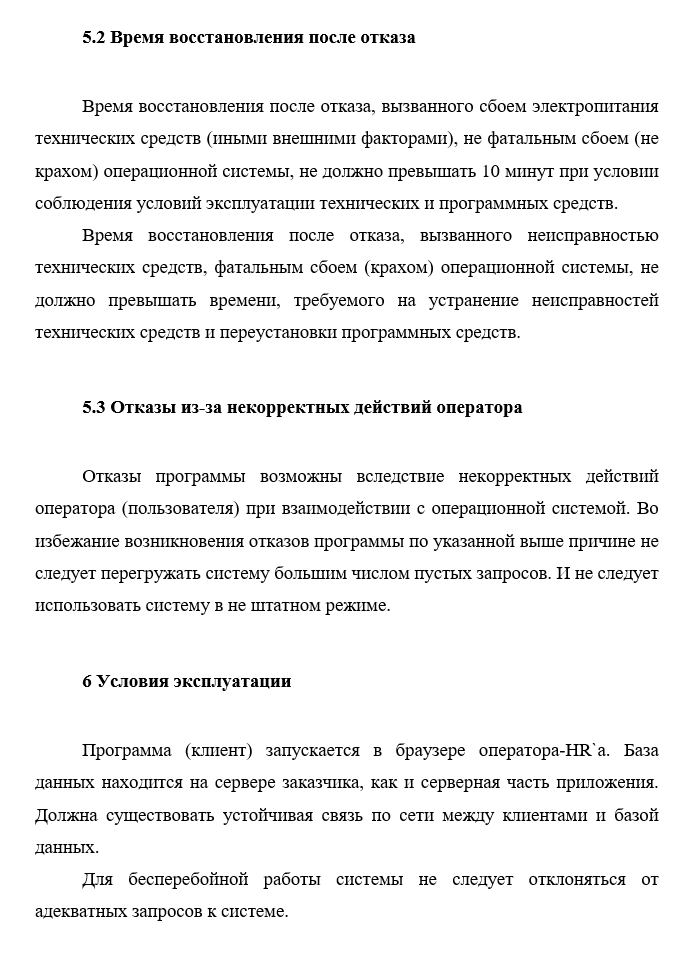
**Продолжение ПРИЛОЖЕНИЯ В**



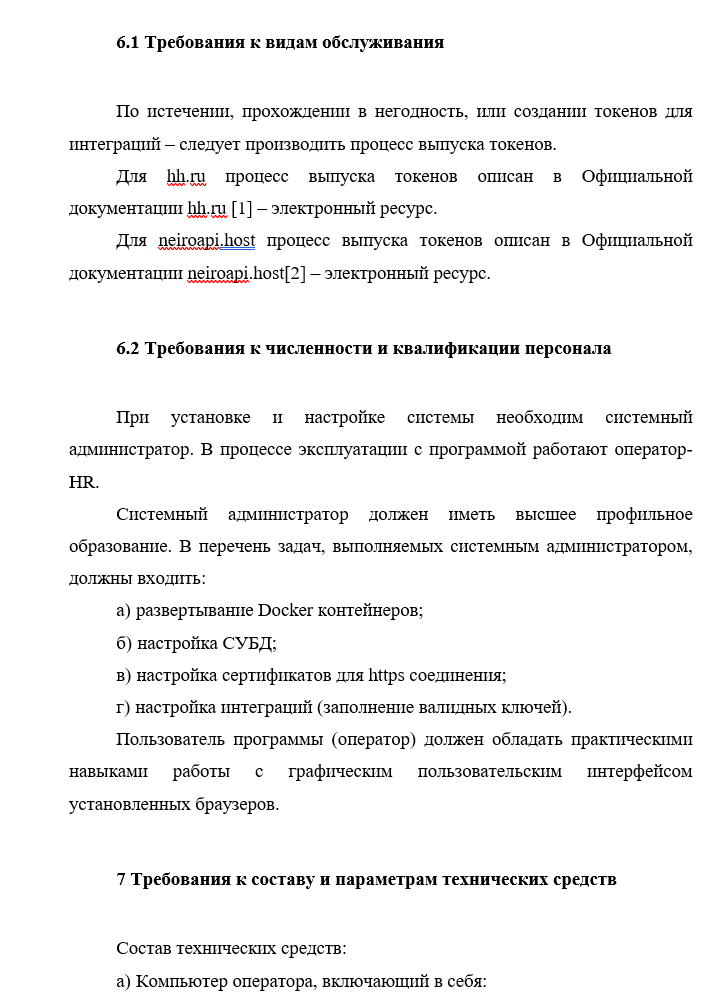
**Продолжение ПРИЛОЖЕНИЯ В**



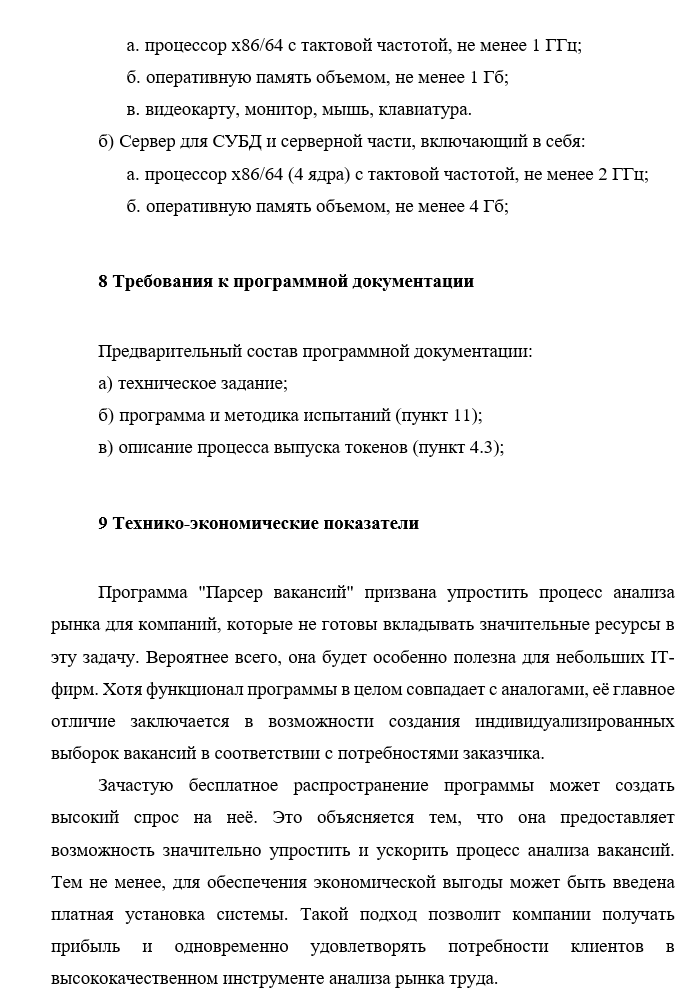
**Продолжение ПРИЛОЖЕНИЯ В**



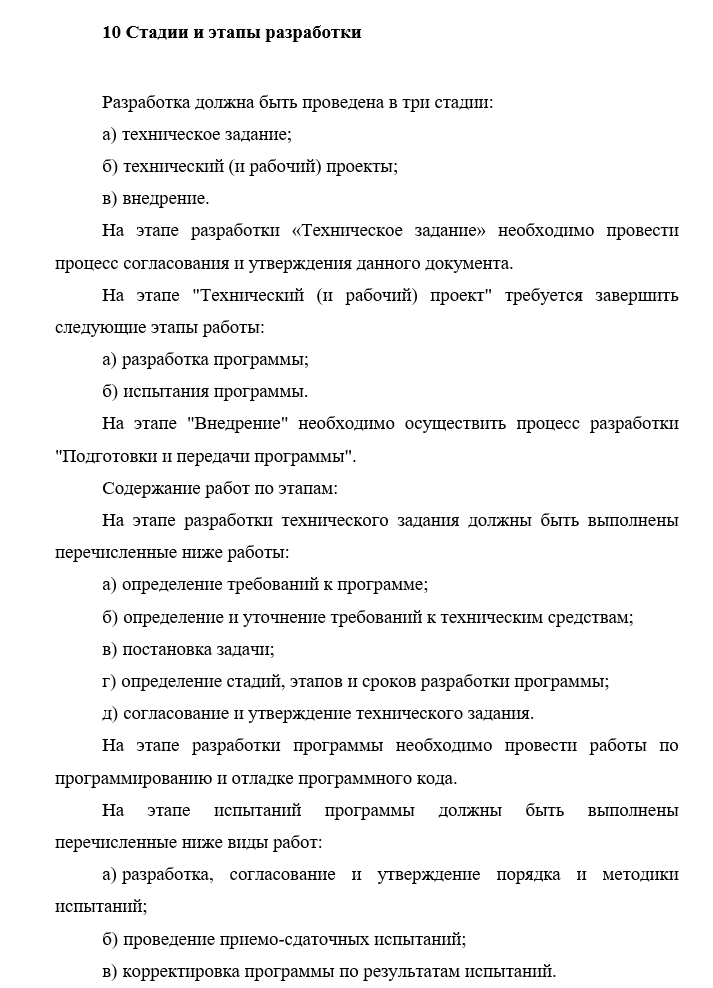
**Продолжение ПРИЛОЖЕНИЯ В**



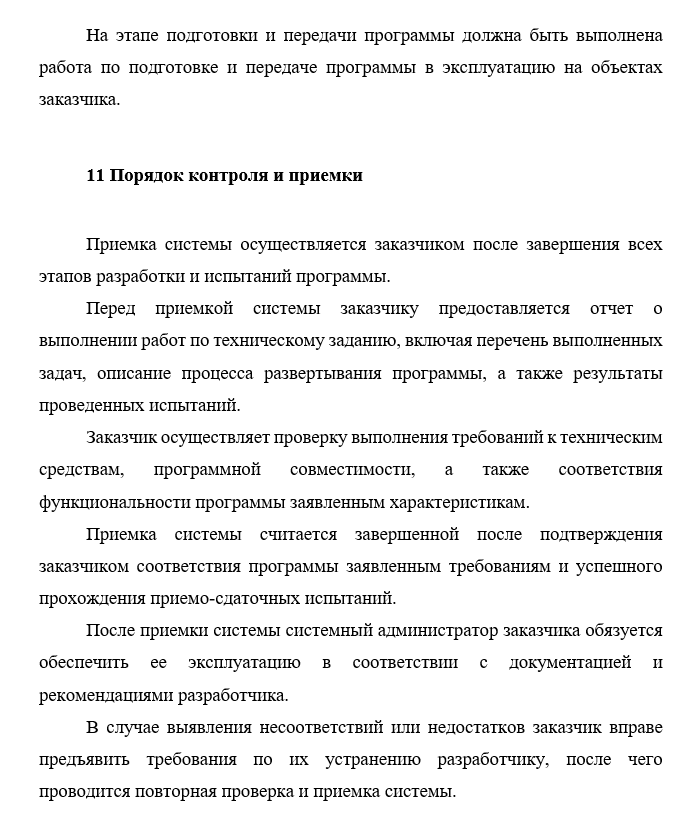
**Продолжение ПРИЛОЖЕНИЯ В**



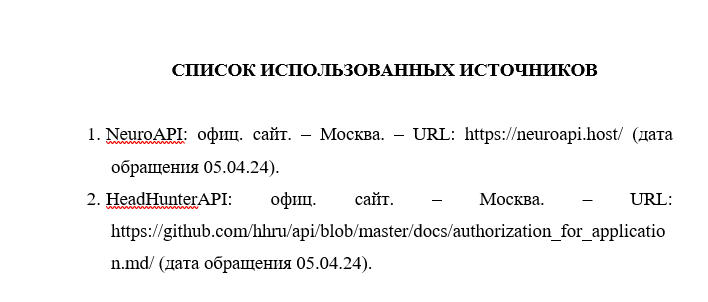
**Продолжение ПРИЛОЖЕНИЯ В**



**Продолжение ПРИЛОЖЕНИЯ В**



**Продолжение ПРИЛОЖЕНИЯ В**



# ПРИЛОЖЕНИЕ Г (обязательное) Содержание xlsx файла (отчета)

Таблица Г.1 – Содержание xlsx файла (отчета)

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Пояснение | Определяется нейросетью | Пример |
| 1 | 2 | 3 | 4 |
| Ссылка на вакансию в hh.ru | Прямая ссылка на страницу вакансии на hh.ru | Нет | https://ekaterinburg.hh.ru/vacancy/90947747 |
| Набор навыков из словаря hh.ru | Ключевые навыки, которые работодатель ожидает у соискателя, взятые из словаря hh.ru | Да | Python, SQL, JavaScript, аналитическое мышление, коммуникабельность |
|  |  |  |  |
| Название вакансии | Официальное название вакансии, с площадки hh.ru | Нет | Специалист по разработке программного обеспечения |
| Грейд | Уровень квалификации, указанный в вакансии | Да | Junior |

**Продолжение ПРИЛОЖЕНИЯ Г**

Продолжение таблицы Г.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Задача и функционал | Описание основных задач и обязанностей, которые предстоит выполнять сотруднику в данной вакансии | Нет | Разработка и поддержка программного обеспечения, участие в проектировании систем, тестирование и оптимизация кода |
| Описание вакансии | Описание вакансии, предоставленное работодателем | Нет | Мы ищем опытного специалиста с углубленными знаниями в области программирования для участия в проектах по созданию высокотехнологичных решений |

**Продолжение ПРИЛОЖЕНИЯ Г**

Продолжение таблицы Г.1

|  |  |  |  |
| --- | --- | --- | --- |
| Требования в вакансии | Опыт работы | Нет | Опыт работы от 3 лет, знание Python и опыт работы с базами данных, высшее техническое образование |
| Наличие ДМС | Информация о наличии медицинской страховки | Да | Да |
| Рабочее время | Информация о наличии рабочем графике | Да | Гибкий график |
| Удаленная работа | Информация о возможности удаленной работы | Нет | Да |
| Результирующий уровень заработной платы | Диапазон заработной платы, который предоставляется для данной вакансии | Нет | 80 000 – 120 000 |

**Продолжение ПРИЛОЖЕНИЯ Г**

Продолжение таблицы Г.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название компании | Название компании, предлагающей вакансию | Нет | ArtSofte |
| ID компании | ID компании с сайта hh.ru | Нет | 1234567 |

# ПРИЛОЖЕНИЕ Д (обязательное) Аналитический запрос

Таблица Д.1 – содержание аналитического запроса

|  |  |
| --- | --- |
| Название поля | Предназначение |
| Текст поиска | Аналогичен «умному поиску» с hh.ru (описание есть в <https://ekaterinburg.hh.ru/article/1175>) |
| Уровень квалификации сотрудника | Поиск производится только среди этого уровня квалификации |
| Регион поиска | Позволяет получить выборку вакансий по определённому региону |
| Наличие ДМС | Предоставляет вакансии только с наличием ДМС |
| Поля поиска | Модифицирует поле «Текст поиска», предоставляя опции: поиск в названии, поиск в названии компании, поиск в описании вакансии |
| Список индустрий | Возможность отфильтровать результат запроса по списку индустрий hh.ru |
| Вакансии только с заработной платой | При включенной опции вакансии без указания заработной платы будут исключены из отчета |
| Тип занятости | Возможность отфильтровать результат запроса по полям: полная занятость, частичная занятость, проектная работа/разовое задание, волонтёрство, стажировка, оформление по ГПХ или по совместительству |
| График работы | Возможность отфильтровать результат запроса по полям: полный день, сменный график, гибкий график, удаленная работа. вахтовый метод |
| Опыт работы | Возможность отфильтровать результат запроса по полям: не имеет значения, нет опыта, от 1 года до 3 лет, от 3 до 6 лет, более 6 лет |

# ПРИЛОЖЕНИЕ Е (обязательное) Сущность аналитического запроса

Таблица Е.1 – содержание аналитического запроса

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Предназначение |
| ID | uuid | Строка для однозначной идентификации сущности |
| search\_field | varchar(255) | Строка для хранения поля поиска |
| qualification\_level | varchar(255) | Строка для хранения уровня квалификации |
| region | json | Объект для хранения сущности региона |
| vmi | boolean | Булевое значение для обозначения наличия ДМС |
| search\_modifier | json | Объект для хранения сущности модификатора поиска |
| industry | json | Объект для хранения сущности индустрии |
| has\_salary | boolean | Булевое значение для обозначения наличия зарплаты в вакансии |
| employment | varchar(255) | Строка для хранения типа занятости |
| schedule | varchar(255) | Строка для хранения графика работы |
| experience | varchar(255) | Строка для хранения опыта работы |
| created\_at\_unix | integer | Unix представление времени создания сущности |

# ПРИЛОЖЕНИЕ Ж (обязательное) Актуализация документов в кодовом представлении



Рисунок Ж.1 – скрипт на обновление запросов