Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной информатики и компьютерных наук

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП

д-р техн. наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.П. Сущенко

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

Разработка веб-приложения для планирования задач и подготовки к техническому собеседованию по направления Java

по направлению подготовки 09.03.03 Прикладная информатика

направленность (профиль) «Прикладная информатика»

Боровских Вадим Борисович

Руководитель ВКР

канд. техн. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.С. Шкуркин

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Автор работы

студент группы № 932003

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Б. Боровских

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Томск, 2024

**ЗАДАНИЕ**

Министерство науки и высшего образования Российской Федерации.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Наименование учебного структурного подразделения

УТВЕРЖДАЮ

Руководитель ООП

д-р техн. наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.П. Сущенко

*подпись*

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающемуся

|  |
| --- |
| Боровских Вадима Борисовичу |

по направлению подготовки 09.03.03 Прикладная информатика, направленность (профиль) «Прикладная информатика»

1 Тема выпускной квалификационной работы

|  |
| --- |
| Разработка веб-приложения для планирования задач и подготовки к техническому |
| собеседованию по направления Java |

2 Срок сдачи обучающимся выполненной выпускной квалификационной работы:

|  |  |  |  |
| --- | --- | --- | --- |
| а) в учебный офис / деканат – |  | б) в ГЭК – |  |

3 Исходные данные к работе:

|  |  |  |  |
| --- | --- | --- | --- |
| Объект исследования – | | разработка веб-приложения для подготовки к собеседованию Java-разработчиков. | |
| Предмет исследования – | | | методы обучения и организации личного времени в процессе подготовки по направлению java-разработки. |
| Цель исследования – | разработка веб-приложения для планирования задач и подготовки к техническому собеседованию. | | |

Задачи:

|  |
| --- |
| 1) Анализ предметной области планирования задач и подготовки к техническому собеседованию.   * Методы подготовки (обучения). * Подходы и методы эффективного планирования задач. |
| 2) Определение требований к функционалу разрабатываемого приложения. |
| 3) Обзор существующих веб-приложений с аналогичным функционалом и простоты использования. |
| 4) Изучение используемых инструментов разработки под требуемый функционал. |
| 5) Проектирование архитектуры веб-приложения. |
| * Описание вариантов использования, схем взаимодействия и пользовательского интерфейса. |
| 6) Разработка веб-приложения. |

Методы исследования:

|  |
| --- |
|  |
|  |

Организация или отрасль, по тематике которой выполняется работа, –

|  |
| --- |
|  |

4 Краткое содержание работы

|  |
| --- |
|  |
|  |

Руководитель выпускной квалификационной работы

|  |  |  |  |
| --- | --- | --- | --- |
| Канд. техн. наук, доцент |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / |  |
| *должность, место работы* |  | *подпись* | *И.О. Фамилия* |

Задание принял к исполнению

|  |  |  |  |
| --- | --- | --- | --- |
| Студент, группа 932003 |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / |  |
| *должность, место работы* |  | *подпись* | *И.О. Фамилия* |

**АННОТАЦИЯ**

**СОДЕРЖАНИЕ**

[ГЛОССАРИЙ 6](#_Toc166784002)

[ВВЕДЕНИЕ 7](#_Toc166784003)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc166784004)

[1.1 Предназначение данного веб-приложения 9](#_Toc166784005)

[1.2 Анализ методов обучения 9](#_Toc166784006)

[1.3 Анализ методов планирования 11](#_Toc166784007)

[2. АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ 15](#_Toc166784008)

[3. АНАЛИЗ ТРЕБОВАНИЙ 19](#_Toc166784009)

[3.1 Функциональные и нефункциональные требования 19](#_Toc166784010)

[3.2 Диаграмма вариантов использования 19](#_Toc166784011)

[3.3 Сценарии вариантов использования 21](#_Toc166784012)

[3.4 Модель предметной области 32](#_Toc166784013)

[3.5 Диаграмма пакетов 36](#_Toc166784014)

[3.6 Диаграмма состояний 37](#_Toc166784015)

[3.7 Диаграмма деятельности 38](#_Toc166784016)

[4. ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ 43](#_Toc166784017)

[4.1 Используемые технологии 43](#_Toc166784018)

[4.1.1 Java 43](#_Toc166784019)

[4.1.2 Spring Boot 43](#_Toc166784020)

[4.1.3 Hibernate 43](#_Toc166784021)

[4.1.4 Maven 44](#_Toc166784022)

[4.1.5 PostgreSQL 44](#_Toc166784023)

[4.1.6 Swagger 44](#_Toc166784024)

[4.1.7 Postman 44](#_Toc166784025)

[4.1.8 Thymeleaf 44](#_Toc166784026)

[4.1.9 JavaScript 45](#_Toc166784027)

[4.2 Архитектура приложения 46](#_Toc166784028)

[4.2.1 Архитектурный паттерн Model-View-Controller (MVC) 46](#_Toc166784029)

[4.2.2 Преимущества использования MVC 46](#_Toc166784030)

[4.2.3 Принцип работы MVC в нашем приложении 46](#_Toc166784031)

[5. РЕАЛИЗАЦИЯ 48](#_Toc166784032)

[5.1 Структура проекта 48](#_Toc166784033)

[5.2 Реализация авторизации и регистрации пользователя 49](#_Toc166784034)

[5.2.1 Авторизация пользователя 49](#_Toc166784035)

[5.2.2 Регистрация пользователя 51](#_Toc166784036)

[5.3 Реализация главной страницы 53](#_Toc166784037)

[5.4 Реализация требований на вакансии 54](#_Toc166784038)

[5.5 Реализация плана изучения 55](#_Toc166784039)

[5.6 Реализация вопросов на вакансии 56](#_Toc166784040)

[5.7 Реализация планировщика 58](#_Toc166784041)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 62](#_Toc166784042)

# **ГЛОССАРИЙ**

1. **Техническое собеседование**: интервью, нацеленное на оценку знаний и навыков в области IT, включая теоретические вопросы по направлению.
2. **Планировщик задач**: инструмент для учета и контроля исполнения поставленных задач, интегрированный в веб-приложение.
3. **Пользователь**: начинающий или имеющий уже, какой-то опыт IT-специалист, планирующий свою деятельность и готовящийся к техническим собеседованиям через использование разрабатываемого веб-приложения.
4. **План:** это общие стратегии обдуманных шагов, сроков и методов для достижения поставленной цели с возможными сроками выполнения.
5. **Задачи:** конкретные шаги, которые необходимо выполнить для достижения цели. Они могут быть измеримыми и определенными, иметь конкретные сроки выполнения и быть частью более общего плана.
6. **Подзадачи:** конкретные задачи, которые необходимо выполнить в рамках основных задач, их выполнение помогает достичь цели, но они не являются основными компонентами плана. Они могут быть менее конкретными и иметь более общий характер, связанный с промежуточными этапами достижения цели.
7. **Подготовка**: процесс подготовки к выполнению задач и достижению целей. Это может включать в себя изучение материала, практику, поиск информации, обратную связь и т.д.
8. **Паттерн**: в программировании это повторяющееся решение общих проблем проектирования и разработки. Паттерны помогают структурировать код и сделать его более понятным и поддерживаемым.
9. **Клиент**: это программа или устройство, которое запрашивает и получает услуги или ресурсы от сервера. В веб-разработке клиент обычно представляет собой веб-браузер, работающий на пользовательском устройстве.
10. **Сервер**: это программа или устройство, предоставляющее услуги или ресурсы клиентам. В веб-разработке сервер обрабатывает запросы от клиентов и предоставляет им запрашиваемые данные или выполняет необходимые действия.

# **ВВЕДЕНИЕ**

Процесс прохождения собеседований является неотъемлемой часть любого человека, который находится в поиске работы. И мало, кто из людей в начале своей профессионально деятельности обладает пониманием, что от него требуется по его направлению и что больше всего интересует работодателей.

Помимо понимания требуемых навыков, нужно уметь правильно планировать свои задачи и развивать компетенции, необходимые для успешного прохождения технических собеседований, становится важным аспектом подготовки к будущей профессиональной деятельности.

В данном контексте разработка веб-приложения, способного эффективно интегрировать функции планирования задач и подготовки к интервью, представляется логичным шагом для современных специалистов в области информационных технологий. Такой инструмент позволит им не только эффективнее управлять своими рабочими задачами, но и осуществлять систематическую подготовку к собеседованиям, что может повысить их шансы на успешное трудоустройство.

Кроме того, важно отметить, что разработка веб-приложения включает в себя не только технические аспекты, но и учет потребностей и предпочтений будущих пользователей. На этапе проектирования приложения необходимо уделить внимание эргономике интерфейса, удобству использования и безопасности личных данных пользователей. Также важно обеспечить поддержку русского языка в пользовательском интерфейсе и использование понятного и интуитивно понятного дизайна. Все эти аспекты также обеспечат успешное внедрение и использование веб-приложения среди целевой аудитории.

**Объект** исследования: разработка веб-приложения для подготовки к собеседованию Java-разработчиков.

**Предмет** исследования: методы обучения и организации личного времени в процессе подготовки по направлению java-разработки.

**Целью** выпускной работы является разработка веб-приложения для планирования задач и подготовки к техническому собеседованию.

Сформулируем следующие **задачи**:

1. Анализ предметной области планирования задач и подготовки к техническому собеседованию.

* Методы подготовки (обучения).
* Подходы и методы эффективного планирования задач.

1. Обзор существующих веб-приложений с аналогичным функционалом и простоты использования.
2. Определение требований к функционалу разрабатываемого приложения.
3. Изучение используемых инструментов разработки под требуемый функционал.
4. Проектирование архитектуры веб-приложения.

* Описание вариантов использования, схем взаимодействия и пользовательского интерфейса.

1. Разработка веб-приложения.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Предназначение данного веб-приложения**

Данное веб-приложение нацелено на изучение Java, включая подготовку к собеседованиям по данному направлению. Приложение позволяет пользователям самостоятельно планировать краткосрочное обучение с помощью планировщика, а также оценивать востребованность определённых знаний на основе статистики вакансий и анализа часто задаваемых вопросов.

**Целями приложения являются:**

* + Формирования опыта постановки задач и сроков их выполнения.
  + Улучшение эффективности процесса обучения.
  + Предоставление статистики вакансий и часто задаваемые вопросы на собеседовании.

## **1.2 Анализ методов обучения**

Чтобы понять, какие действия будут предприняты в рассматриваемом приложении, следует рассмотреть существующие подходы в обучении и определить ориентиры для создания этого приложения.

В течение процесса обучения и освоения новых знаний различными индивидами были сформированы различные методы передачи знаний. Представлено множество классификаций, основанных на одном или нескольких признаках. Каждый автор приводит аргументы для обоснования своей классификационной модели. Рассмотрим два метода, первый из которых классифицирует по источнику передачи и характеру восприятия информации [4]:

* пассивное восприятие — слушают и смотрят (рассказ, лекция, объяснения; демонстрационный);
* активное восприятие — работа с книгой, наглядными источниками; лабораторный метод.

И второй метод организации и осуществления учебно-познавательной деятельности [5]:

* + словесные методы (книги, рассказ, лекции, беседы);
  + наглядные методы (иллюстрации, демонстрации);
  + практические методы (практические занятия, курсовые и дипломные работы);
  + репродуктивные и проблемно-поисковые (от частного к общему, от общего к частному);
  + методы самостоятельной работы и работы под руководством преподавателя.

**Первый метод** подчёркивает различие в восприятии информации, исходя из её источника. Он делит взаимодействие человека с информацией на два основных типа: пассивное и активное восприятие.

Пассивное восприятие характеризуется как принятие информации через слушание и наблюдение (например, при прослушивании рассказов, лекций или просмотре демонстраций). Этот тип восприятия считается менее эффективным для глубокого усвоения материала.

Активное восприятие включает в себя непосредственное участие и работу с информацией (работа с книгой, использование наглядных материалов, лабораторные работы), что способствует более глубокому усвоению материала.

**Второй метод** акцентирует внимание на разнообразии подходов к организации учебно-познавательной деятельности, которые включают:

Словесные методы основаны на устном и письменном обмене информацией между преподавателем и учащимися.

Наглядные методы нацелены на визуальное обучение и облегчают понимание сложных концепций через визуальное восприятие.

Практические методы направлены на практическое применение знаний и навыков, обучая людей различным реальным сценариям.

Репродуктивные и проблемно-поисковые методы, направленные на развитие аналитического мышления от частного к общему и наоборот.

Методы самостоятельной работы и работы под руководством преподавателя, способствующие развитию самостоятельности и критического мышления.

Эти методы обучения могут использоваться в комбинации для достижения максимальной эффективности в образовательном процессе, учитывая индивидуальные потребности и стиль обучения учащихся.

Оба метода подчёркивают важность подбора соответствующего способа взаимодействия с информацией, указывая на то, что выбор метода может существенно влиять на эффективность усвоения информации и качество обучения в целом. Учитывая данные методы, задачей будет стоять разработка веб-приложения, которое будет эффективно способствовать самостоятельному изучению благодаря активным и пассивным методам обучения, а также стимулировать пользователя к развитию навыков работы с информацией и к самостоятельному познанию.

## **1.3 Анализ методов планирования**

После рассмотрения существующих методов обучения, следует рассмотреть методы планирования, применяемые в планировщиках задач. Для понимания наиболее эффективных подходов и определения ориентиров важно учитывать разнообразие существующих методик.

Во-первых, необходимо определить, зачем требуется планирование. В повседневной жизни перед человеком стоят многочисленные задачи, каждая из которых имеет различную важность. Без адекватного планирования некоторые задачи могут быть упущены из-за нехватки времени, что приводит к пропуску сроков, тревожности и ухудшению общего состояния. Для этого существуют разнообразные инструменты для планирования времени, которые помогают эффективно распределять его.

Планирование — это метод рационального распределения времени и последовательного выполнения задач в зависимости от их важности, срочности и ресурсов, требуемых для выполнения.

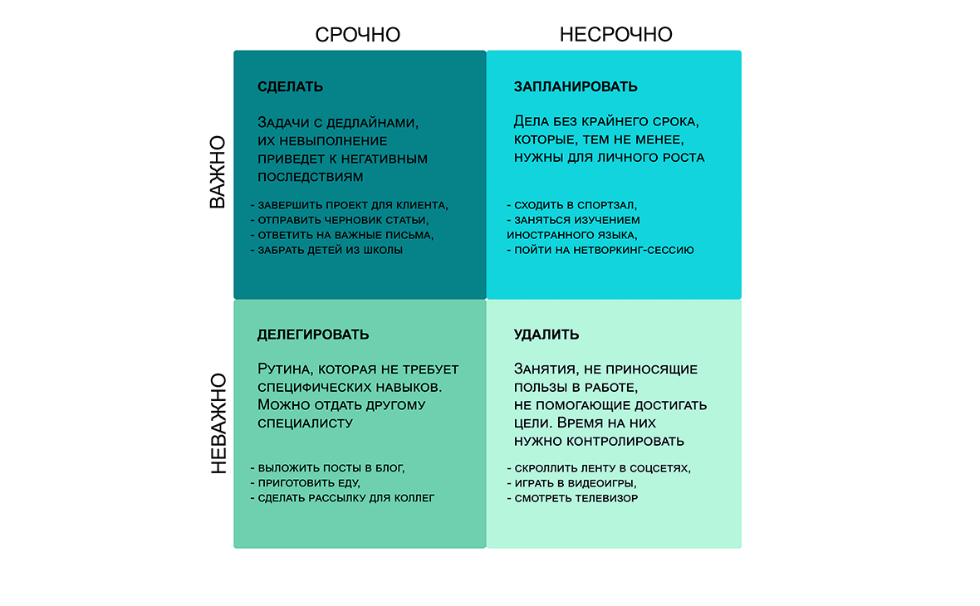
Рассмотрим наиболее популярные методы планирования:

* Матрица Эйзенхауэра
* Пирамида Бенджамина Франклина

Разберем что представляет собой **матрица Эйзенхаура**. Суть метода Эйзенхауэра заключается в предположении, что каждая задача может быть измерена по двум критериям: важности и срочности. За счет этого можно структурировать задачи и **запланировать** **свой** **день**, определив наиболее важные задачи, которые необходимо выполнить в отведенный срок. В основе метода лежит идея **распределение задач по четырем категориям**:

* *Самые важные и срочные задачи* (внезапные проблемы, неотложные задачи проекта, курсовая работа) - задачи с высокой важностью, которую необходимо выполнить в кротчайшие сроки.
* *Важные, но несрочные задачи* (планирование проектов, обучение\развитие навыков) – задачи без срока, но важные для человека.
* *Неважные, но срочные задачи* (ответы на электронные письма, готовка, уборка) – задачи, которые можно, кому-то передать, чтобы освободить время под важное.
* *Неважные и несрочные задачи* (чтение новостей, просмотр видео, просмотр телевизора) – задачи, которые не приносят пользу, следует или убрать или сократить их время.

Эйзенхауэр советовал начинать с важных и срочных задач, а оставшиеся задачи размещать в соответствующих категориях. Так можно использовать наиболее продуктивное время на выполнение важных задач, а также управлять своим расписанием более эффективно, сэкономить время и сократить стресс. На рисунке 1 изображена матрица Эйзенхауэра.



*Рисунок 1 – Пример матрицы Эйзенхауэра*

Разобрав матрицу Эйзенхауэра, можно сделать выводы о **положительных сторонах**:

* **Простота применения:** Одно из главных преимуществ заключается в легкости понимания и использования.
* **Повышение эффективности:** помогает эффективно распределять время между задачами и ставить приоритеты исходя из срочности и важности.
* **Улучшает принятия решений:** матрица обучает методике принятия взвешенных решений, позволяя лучше анализировать и оценивать аспекты жизни и работы.
* **Гибкость методики:** высокая адаптация под индивидуальные потребности пользователей, как для профессиональной, так и для личного времени.

Также рассмотрим, какие есть **недостатки данной матрицы**:

* **Субъективность оценки:** выделение задач зависит от индивидуального взгляда на важность и срочность, что может приводить к ошибке в распределении приоритетов.
* **Ориентированность на краткосрочное планирования:** матрица ориентирована на относительно краткосрочное планирование, что приводит к проблемам в управлении долгосрочными проектами.
* **Переусердствовали с процессом:** имеется риск того, что чрезмерное внимание к классификации и анализу задач может отвлечь от их непосредственного выполнения.

**Что можем улучшить** – это комбинировать с другими методами для более длительного планирования и постановок целей, а также адаптация метода для цифрового пространства, чтобы время между планирование и началом выполнения задач не растягивалось.

Теперь рассмотрим **пирамиду Бенджамина Франклина**.

Пирамида Бенджамина Франклина, хотя и не так известна, как матрица Эйзенхаура, представляет собой уникальный подход к самоорганизации и поэтапного планирования от глобальных жизненных ценностей до плана на день, который служит воплощению этих ценностей. Данный метод основывается на идее составления списка дел, которые Франклин считал необходимыми для успеха в жизни, и последующего строгого самоконтроля за их соблюдением. На рисунке 2 изображен пример пирамиды.



Р*исунок 2 – Пирамида Бенджамина Франклина*

**Главные жизненные ценности** — это внутренний кодекс жизни человека, который будет с ним в независимости от обстоятельств и нацелен чтобы сделать себя лучше.

**Глобальная жизненная цель** — что напрямую связано с жизненными ценностями и отвечает на вопросы о том, как хотелось бы прожить свою жизнь и каких результатов достичь. Например, стать известным в какой-то области, реализоваться в профессии, основать компанию или создать лекарство от болезни.

**Генеральный план достижения целей** — поэтапный план достижения каждой глобальной цели, который включает конкретные шаги. Например, если цель — реализация в профессии, то шаги могут быть такими: получить высшее образование, найти стажировку, войти в основной штат и т.д.

**Долгосрочный план** на 3 года включает ближайшие шаги, которые можно выполнить за это время. Подробно планировать не стоит, но сделать план и сверятся с ним, постепенно дополняя и корректируя в соответствии с обстоятельствами.

**Среднесрочный план(краткосрочный).** На этом уровне фокус смещается на более мелкие и скоротечные задачи, которые нужно выполнить в течение нескольких месяцев или года, чтобы продвинуться к долгосрочным целям. Стоит указывать дедлайны для каждой задачи, чтобы мотивировать и не терять время.

**Ежедневное или еженедельное планирование.** Этот уровень включает в себя более детальное планирование задач на предстоящую неделю. Это способствует организованному использованию времени и обеспечивает продвижение к более крупным целям. На самом нижнем уровне находится план на день, где каждая задача детально спланирована. Это важный элемент, так как ежедневные дела должны быть в гармонии с общими целями и ценностями человека, и приближать к его глобальным целям.

В **достоинствах** наблюдается **общая целостность** подхода и взаимосвязь всех уровней, а также возможность самоконтроля и **фокусировка** **на** **важных** задачах.

В тоже время в **недостатки** можно отнести **сложность** **внедрения** и **время** **затратность**, а также риск перегрузки в попытках контролировать и планировать столько уровней.

В целом, пирамида Бенджамина Франклина демонстрирует ценность структурированного планирования и самодисциплины в достижении личных и профессиональных успехов, а также предлагают выработать свои ценности, что может положительно сказаться на жизни человека.

Оба подхода имеют как достоинства, так и недостатки. Учитывая это можно попробовать совместить оба метода, метод Эйзенхаура хорошо подходит для краткосрочного планирования и выделение важных задач, в то время как пирамида помогает внести в нашу жизнь долгосрочное планирование и развитию жизненных черт человека.

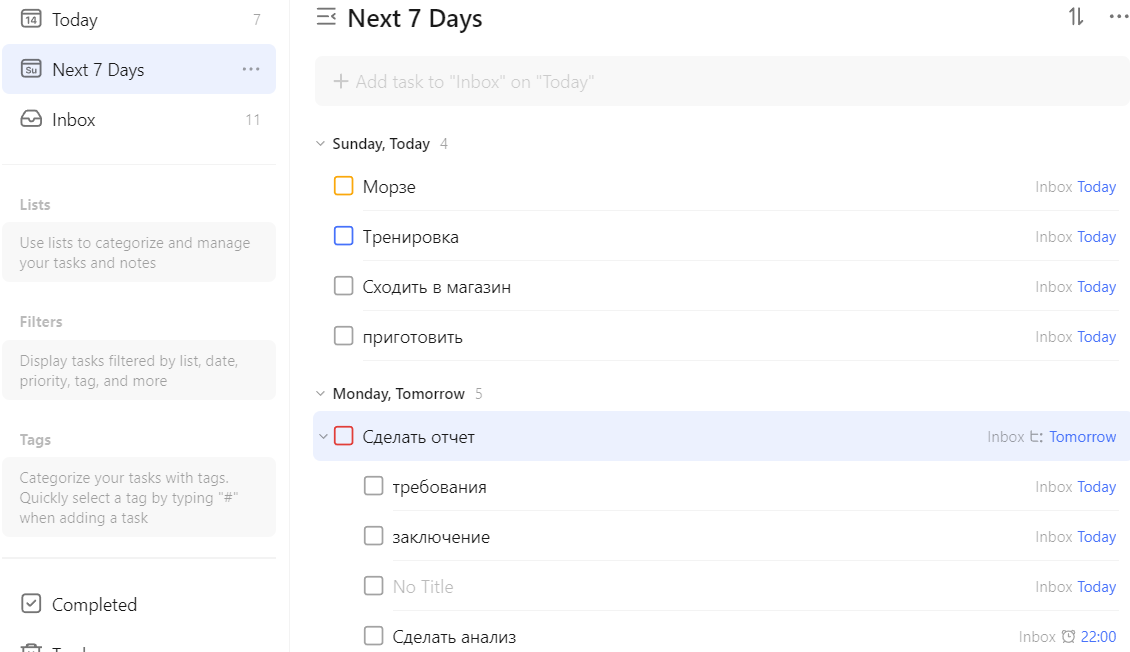
# **АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИ****Й**

Проведя анализ методов обучения и планирования, следует рассмотреть приложения для планирования и подготовки. Необходимо проанализировать наиболее популярные приложения в данной области, такие как YouGile, TickTick и Trello.

YouGile и Trello применяются в современной разработке для планирования задач команд разработчиков, тогда как TickTick используется для планирования краткосрочных задач в повседневной жизни. Анализ указанных инструментов позволит выявить их основные преимущества, недостатки, функциональные возможности и области применения, что окажет влияние на дальнейший выбор оптимального инструмента для целей настоящего исследования.

Начнем с анализа простого планировщика задач TickTick. На рисунке демонстрируются основные функциональные возможности данного приложения**:**

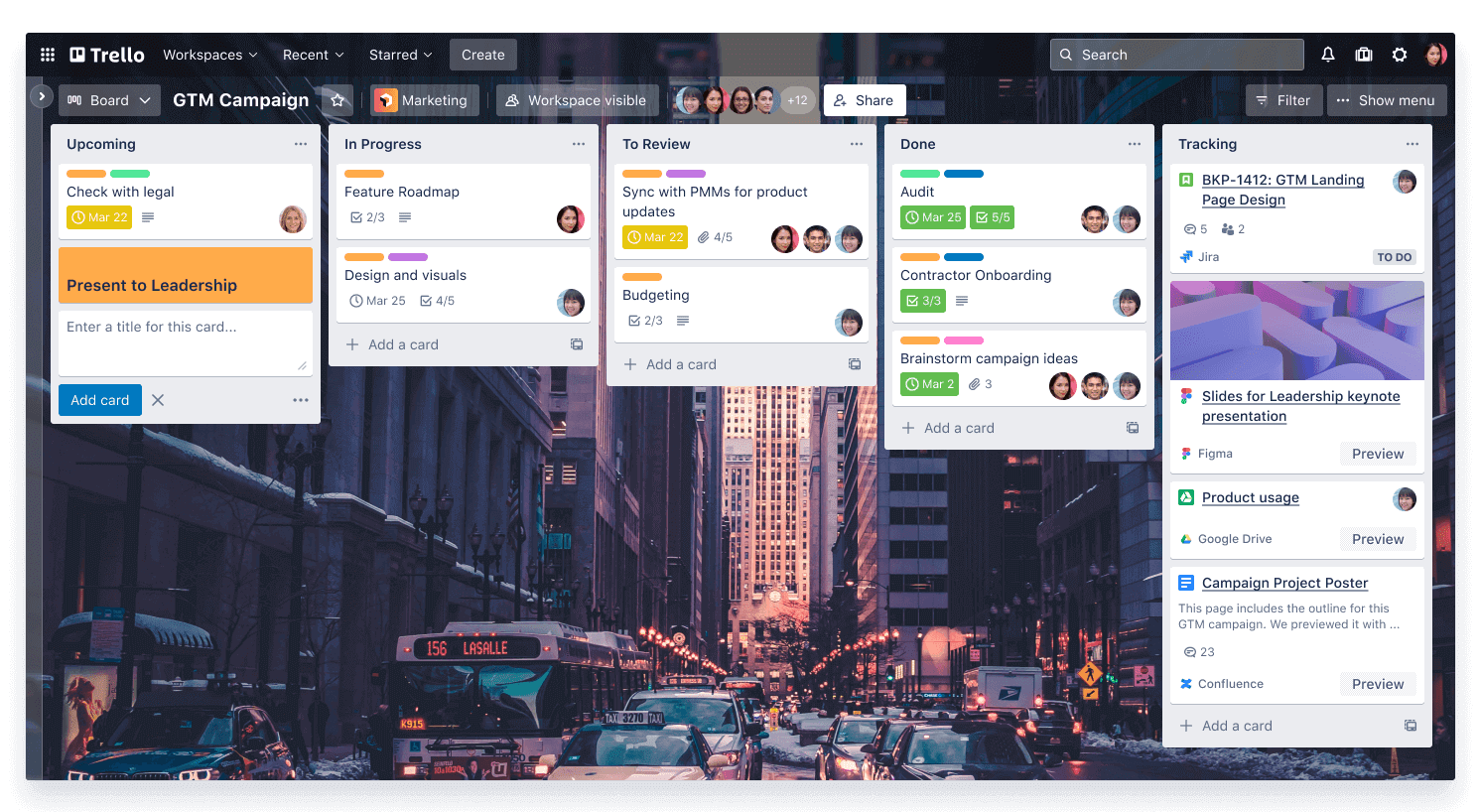
* Постановка задач со сроком выполнения посуточно\неделю.
* Ставить задачи и разбивать на подзадачи
* Установка приоритета задач на 4 типа по цветам: **красный** – самый срочный, **желтый** – средней срочный, **синий** – низко срочный и **бесцветный** – необязательный.



*Рисунок 3 – Интерфейс приложения «Ticktick»*

Рассмотрим самый простой планировщик задач Trello. На рисунке демонстрируется основной **функциональные** **возможности**:

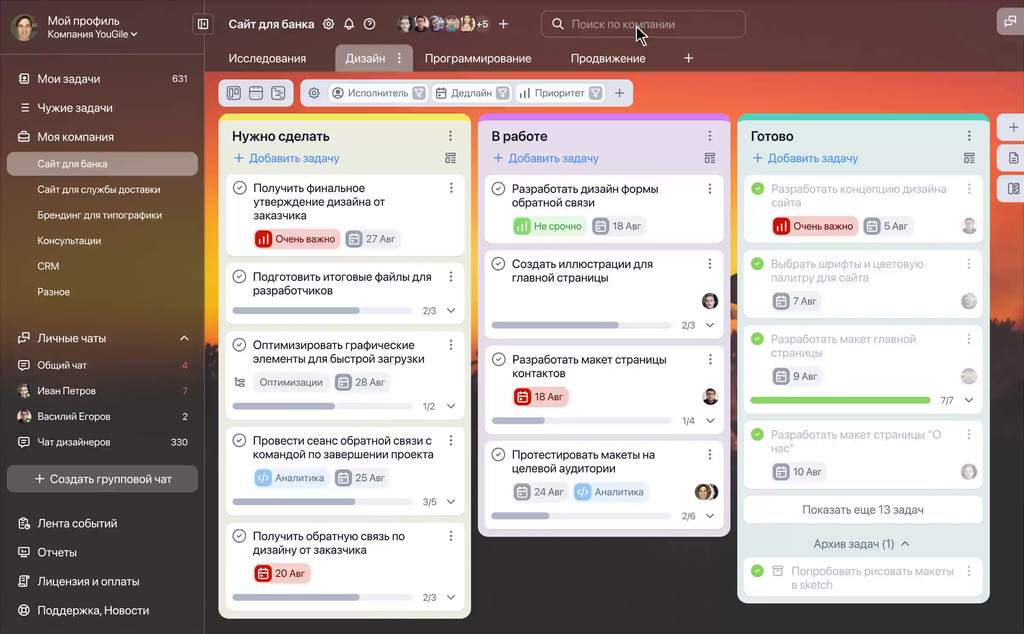
* **Визуализация**: канбан-доски для визуального отображения проектов и задач.
* **Гибкость**: подходит для различных типов проектов и команд любого размера.
* **Расширяемость**: большое количество доступных плагинов и интеграций для расширения функциональности.



*Рисунок 4 – Интерфейс приложения «Trello»*

Рассмотрим самый простой планировщик задач Yougile. На рисунке демонстрируется основной **функциональные** **возможности**:

* **Гибкость**: поддерживает методологии Agile и Scrum, что делает его идеальным для команд разработчиков.
* **Интерактивность**: интерактивная доска задач с возможностью перетаскивать элементов.
* **Интеграция**: легко интегрируется с различными инструментами разработки и коммуникации.



*Рисунок 5 – Интерфейс приложения «Yougile»*

После рассмотрения основного функционала данных приложений, будем анализировать их с учетом определенных критериев.

Таблица 1 – Сравнительная таблица существующих решений

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии | Ticktick | YouGile | Trello |
| Доступность в России | Доступно | Доступно | Недоступно |
| Простота взаимодействия | Высокая | Средняя | Средняя |
| Гибкость | Подходит для индивидуального использования | Подходит для команд | Подходит для команд |
| Приоритет задач | Есть | Есть | Есть |
| Управление проектами | Ограниченное | Расширенное | Расширенное |
| Интеграция с другими сервисами | Ограниченная | Широкая | Широкая |
| **Функции для подготовки к собеседованию** | Нет | Нет | Нет |

Исходя из представленной таблицы сравнения, можно сделать вывод, что приложения YouGile и Trello ориентированы преимущественно на командную работу, предоставляя расширенные возможности управления проектами и интеграции с другими сервисами. В то же время, TickTick выделяется своей доступностью и простотой взаимодействия, что делает его идеальным выбором для индивидуального использования.

# **АНАЛИЗ ТРЕБОВАНИЙ**

## **3.1 Функциональные и нефункциональные требования**

Исходя из анализа методов и приложений в данной области, были выведены конкретные функциональные и нефункциональные требования к создаваемому приложению.

**Функциональные требования**:

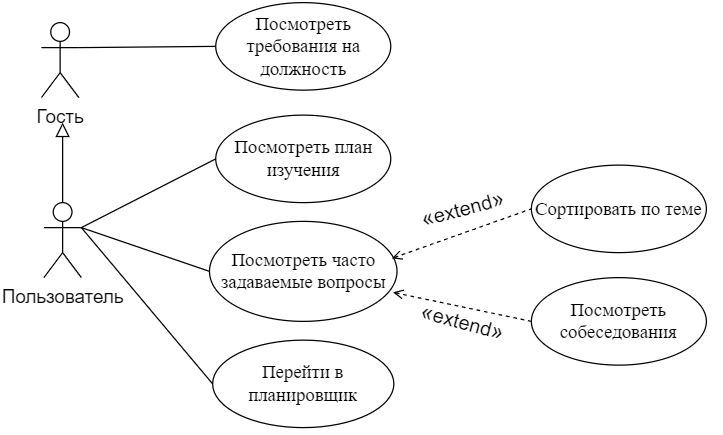
1. Добавление и редактирование задач.
2. Маркировка статуса задач.
3. Анализ требований по Java.
4. База данных вопрос и ссылок на собеседования

**Нефункциональные требования**:

1. Русская версия пользовательского интерфейса
2. Простой и понятный интерфейс.
3. Безопасность личных данных пользователя.
4. Разработка на Java.
5. Использование PostgreSQL в качестве СУБД.

## **3.2 Диаграмма вариантов использования**

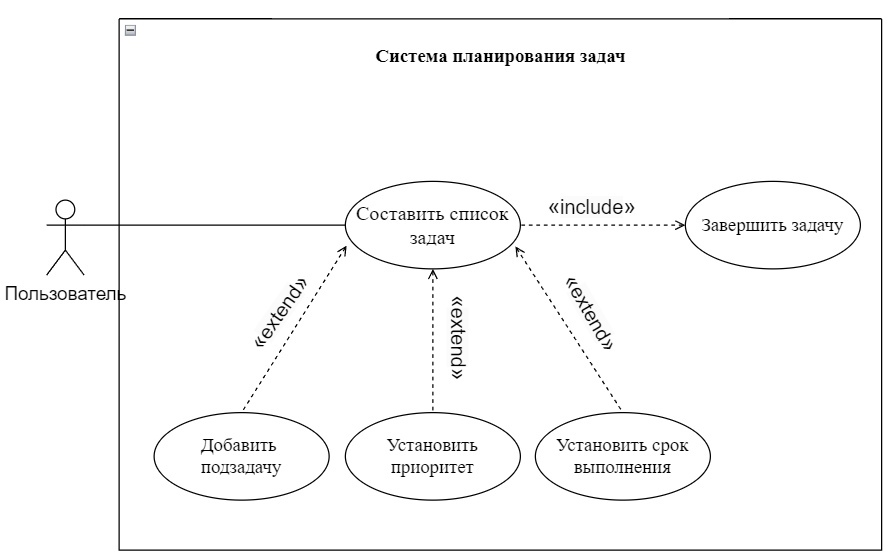
В процессе анализа и выявления требований к разрабатываемому приложению был определен следующий набор актёров: Гость, Пользователь и Администратор. Функциональные требования, устанавливаемые для приложения, были подробно описаны в диаграммах вариантов использования [1,7], представленных ниже.



*Рисунок 6 – Диаграмма вариантов использования веб-приложения*

Прецеденты диаграммы вариантов использования веб-приложения:

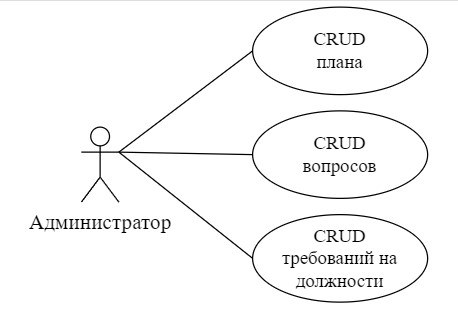
1. *Посмотреть требования на должность:* Гость и Пользователь имеет возможность ознакомления со статистикой требований к вакансиям.
2. *Посмотреть план изучения:* Пользователь получают доступ к рекомендациям по изучению тем и полезным ресурсам.
3. *Посмотреть часто задаваемы вопросы:* Пользователь переходит на страницу с часто задаваемыми вопросами и изучает наиболее распространённые вопросы.
   1. *Сортировать по теме:* Пользователь фильтрует вопросы по интересующим темам.
   2. *Посмотреть собеседования:* Пользователь может просматривать записи собеседований для подготовки.
4. *Перейти в планировщик:* Пользователь может составлять персональный список задач, перейдя на соответствующую страницу.



*Рисунок 7 – Диаграмма вариантов использования планировщика*

Прецеденты диаграммы вариантов использования планировщика:

1. *Составить список задач:* Пользователь может составить свой список задач для изучения по конкретной теме или вопросу.
   1. *Добавить подзадачу:* Пользователь может добавить подзадачу, тем самым разбив задачу на подпункты.
   2. *Установить приоритет:* Пользователь может дать приоритет важность задачи.
   3. *Установить срок выполнения:* Пользователь может добавить дату, до которой нужно сделать.
   4. *Завершить задачу:* Пользователь может завершить задачу.



*Рисунок 8 – Диаграмма вариантов использования панель администратора*

Прецеденты диаграммы вариантов использования панели администратора:

1. *CRUD плана:* Администратор имеет возможность вносить изменения в план обучения.
2. *CRUD вопросов:* Администратор может вносить изменения в базу вопросов.
3. *CRUD требований на должности:* Администратор может редактировать требования к должности.

## **3.3 Сценарии вариантов использования**

Ниже представлены сценарии использования системы, в которых выделены различные роли актеров: Гость, Пользователь и Администратор. Общее количество - 14 сценариев использования:

1. Посмотреть требования на должность.
2. Посмотреть план изучения.
3. Посмотреть часто задаваемые вопросы.
4. Сортировать по теме.
5. Посмотреть собеседования.
6. Перейти в планировщик.
7. Составить список задач.
8. Добавить подзадачу.
9. Установить приоритет.
10. Установить срок выполнения.
11. Завершить задачу.
12. CRUD плана.
13. CRUD вопросов.
14. CRUD требования на должности.

Дальше рассмотрим каждый из этих прецедентов подробно:

**Таблица 2.1** – Сценарий варианта использования «Посмотреть требования на должность»

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Посмотреть требования на должность. | |
| **Цель** | |
| Перейти с главной страницы к странице с требований. | |
| **Актёр** | |
| Гость, Пользователь. | |
| **Предусловие** | |
| Пользовать находится на главной странице. | |
| **Сценарий** | |
| 1. Нажимает на название «Требования на должность». | 2. Загружает страницу с требованиями. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь перешел на страницу с требованиями. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу» | |

**Таблица 2.2** – Сценарий варианта использования «Посмотреть план изучения»

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Посмотреть план изучения. | |
| **Цель** | |

Окончание таблицы 2.2.

|  |  |
| --- | --- |
| Перейти с главной страницы к странице с планом. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на главной странице. | |
| **Сценарий** | |
| 1.Нажимает на название «План изучения». | 2.Загружает страницу с планом изучения. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь перешел на страницу с планом. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу» | |

**Таблица 2.3** – Сценарий варианта использования «Посмотреть часто задаваемые вопросы»

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Посмотреть часто задаваемые вопросы. | |
| **Цель** | |
| Перейти с главной страницы к странице с вопросами. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на главной странице. | |
| **Сценарий** | |
| 1. Нажимает на название «Вопросы на собеседованиях». | 2. Загружает страницу с вопросами. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь перешел на страницу с вопросами. | |
| **Точки расширения** | |

Окончание таблицы 2.3.

|  |
| --- |
| Возможен переход к прецеденту «Вернуться в главное меню», «Сортировать по теме» или «Посмотреть собеседования» |

**Таблица 2.4** – Сценарий варианта использования «Сортировать по теме»

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Сортировать по теме. | |
| **Цель** | |
| Отсортировать по определенной теме. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице вопросов. | |
| **Сценарий** | |
| 1. Нажимает на кнопку с темами и выбирает нужную. | 2. Обновляет страницу и выводит вопросы только по определённой теме. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователю вывелись темы, которые его интересуют. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться в главное меню», «Посмотреть часто задаваемые вопросы» или «Посмотреть собеседования» | |

**Таблица 2.5** – Сценарий варианта использования «Посмотреть собеседования»

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Посмотреть собеседования. | |
| **Цель** | |
| Перейти со страницы вопросов на просмотр собеседования. | |
| **Актёр** | |
| Пользователь. | |

Окончание таблицы 2.5.

|  |  |
| --- | --- |
| **Предусловие** | |
| Пользовать находится на странице вопросов. | |
| **Сценарий** | |
| 1. Нажимает на 1 из ссылок с собеседование. | 2. Загружается собеседование. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь перешел запись собеседования. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться в главное меню», «Посмотреть часто задаваемые вопросы» или «Сортировать по теме». | |

**Таблица 2.6** – Сценарий варианта использования «Перейти в планировщик».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Перейти в планировщик. | |
| **Цель** | |
| Перейти с главной страницы к странице с планировщиком. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на главной странице. | |
| **Сценарий** | |
| 1.Нажимает на название «Планировщик». | 2.Загружает страницу с планировщиком. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь перешел на страницу с планировщиком. | |
| **Точки расширения** | |

Окончание таблицы 2.6.

|  |
| --- |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Добавить подзадачу», «Установить приоритет», «Установить срок выполнения», «Завершить задачу» |

**Таблица 2.7** – Сценарий варианта использования «Составить список задач».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Перейти в планировщик. | |
| **Цель** | |
| Составить список задач для выполнения. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице планировщика. | |
| **Сценарий** | |
| 1. Вводит требуемы поля, для создания задачи.  2. После заполнения требуемых полей, нажимает на кнопку создать. | 3. Связывает созданную задачу с профилем пользователя.  4. Обновляет страницу и показывает пользователю созданную задачу. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь увидел перед собой составленный список задач. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Добавить подзадачу», «Установить приоритет», «Установить срок выполнения», «Завершить задачу» | |

**Таблица 2.8** – Сценарий варианта использования «Добавить подзадачу».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |

Окончание таблицы 2.8.

|  |  |
| --- | --- |
| Перейти в планировщик. | |
| **Цель** | |
| Добавить к имеющимся задачам, подзадачу. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице планировщика. | |
| **Сценарий** | |
| 1.Нажимает кнопку для создания подзадачи.  2. Вводит все обязательные поля.  3. Нажимает на кнопку создать. | 4. Связывает созданную задачу с id задачи.  5. Обновляет страницу и выводит список задач. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь добавил подзадачу. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Установить приоритет», «Установить срок выполнения», «Завершить задачу» | |

**Таблица 2.9** – Сценарий варианта использования «Установить приоритет».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Перейти в планировщик. | |
| **Цель** | |
| Добавить к имеющимся задачам или подзадачам, один из трех приоритетов. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице планировщика. | |
| **Сценарий** | |

Окончание таблицы 2.9.

|  |  |
| --- | --- |
| 1.Нажимает на кнопку редактировать задачу.  2. Выбирает один из трех приоритетов («Низкий», «Средний», «Высокий»).  3. Нажимает на кнопку «Обновить». | 4. Обновляет информацию о задаче.  5.Выводит обновленные данные пользователю. |
| Альтернативный сценарий | |
| Нет | |
| Постусловие | |
| Пользователь установил приоритет задаче. | |
| Точки расширения | |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Добавить подзадачу», «Установить срок выполнения», «Завершить задачу» | |

**Таблица 2.10** – Сценарий варианта использования «Установить срок выполнения».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Перейти в планировщик. | |
| **Цель** | |
| Добавить к имеющимся задачам или подзадачам, срок выполнения. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице планировщика. | |
| **Сценарий** | |
| 1. Нажимает на кнопку редактировать задачу.  2. Выбирает срок выполнения  3. Нажимает на кнопку «Обновить | 4. Обновляет информацию о задаче.  5.Выводит обновленные данные пользователю. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь установил срок выполнения задачи. | |

Окончание таблицы 2.10

|  |
| --- |
| **Точки расширения** |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Добавить подзадачу», «Установить приоритет», «Завершить задачу» |

**Таблица 2.11** – Сценарий варианта использования «Завершить задачу».

|  |  |
| --- | --- |
| **Пользователь** | **Приложение** |
| **Название ВИ** | |
| Завершить задачу. | |
| **Цель** | |
| Выбрать задачу или подзадачу, которую стоит пометить как выполненная. | |
| **Актёр** | |
| Пользователь. | |
| **Предусловие** | |
| Пользовать находится на странице планировщика. | |
| **Сценарий** | |
| 1.Пользователь выбирает задачу.  2. Задача помечается, как завершенная. | 3. Приложения устанавливает время завершения задачи.  4. Пользователю отображается задача, как завершённая. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Пользователь пометил задачу или подзадачу, как завершенная. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу», «Составить список задач», «Добавить подзадачу», «Установить приоритет», «Установить срок выполнения», «Составить список задач» | |

**Таблица 2.12** – Сценарий варианта использования «CRUD плана»

|  |  |
| --- | --- |
| **Администратор** | **Приложение** |
| **Название ВИ** | |
| CRUD плана. | |

Окончание таблицы 2.12.

|  |  |
| --- | --- |
| **Цель** | |
| Обновить информацию в плане. | |
| **Актёр** | |
| Администратор. | |
| **Предусловие** | |
| Администратор находится на странице плана. | |
| **Сценарий** | |
| 1. Нажимает на панель редактирования плана. 2. Дается возможность CRUD тем в плане.  3. Сохранить изменения. | 4.Обновляет информацию о теме.  5. Выводится обновлённая информация. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Администратор изменил информацию в плане. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу» | |

**Таблица 2.13** – Сценарий варианта использования «CRUD вопросов»

|  |  |
| --- | --- |
| **Администратор** | **Приложение** |
| **Название ВИ** | |
| CRUD вопросов. | |
| **Цель** | |
| Обновить информацию о вопросах. | |
| **Актёр** | |
| Администратор. | |
| **Предусловие** | |
| Администратор находится на странице вопросов. | |
| **Сценарий** | |
| 1. Нажимает на панель редактирования вопросов. 2. Дается возможность CRUD вопросов. | 4.Обновляет информацию о вопросах.  5. Выводится обновлённая информация. |

Окончание таблицы 2.13.

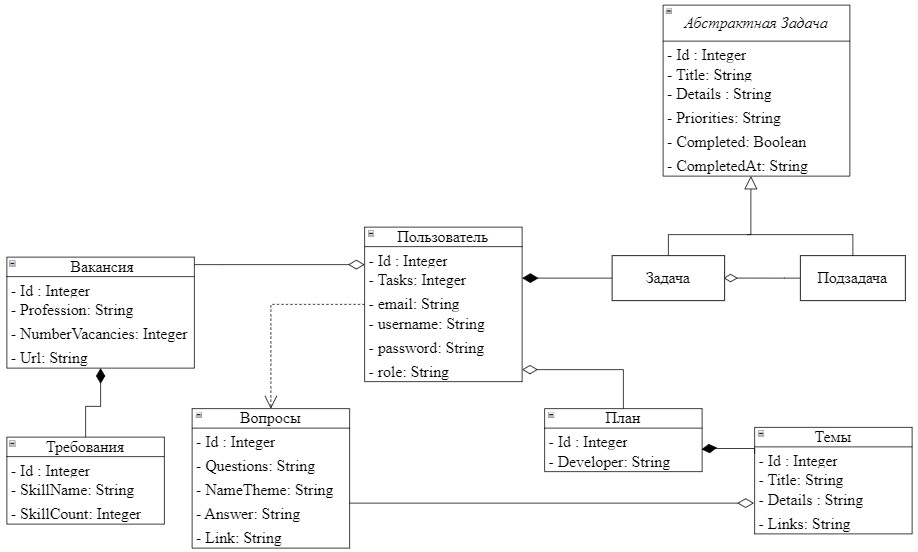
|  |  |
| --- | --- |
| 3. Сохранить изменения. |  |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Администратор изменил информацию о вопросах. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу» | |

**Таблица 2.14** – Сценарий варианта использования «CRUD требований»

|  |  |
| --- | --- |
| **Администратор** | **Приложение** |
| **Название ВИ** | |
| CRUD требований. | |
| **Цель** | |
| Обновить информацию о требованиях. | |
| **Актёр** | |
| Администратор. | |
| **Предусловие** | |
| Администратор находится на странице требований. | |
| **Сценарий** | |
| 1. Нажимает на панель редактирования требований. 2. Дается возможность CRUD требований.  3. Сохранить изменения. | 4.Обновляет информацию о требованиях.  5. Выводится обновлённая информация. |
| **Альтернативный сценарий** | |
| Нет | |
| **Постусловие** | |
| Администратор изменил информацию о требованиях. | |
| **Точки расширения** | |
| Возможен переход к прецеденту «Вернуться на главную страницу» | |

## **3.4 Модель предметной области**

В данном веб-приложении используется девять основных классов предметной области, которые способствуют организации и структурированию информации для эффективного обучения и подготовки к собеседованиям по Java. Описание и назначение этих классов приведено ниже.



*Рисунок 9 – Модель Предметной области*

**Описание классов, их полей и взаимодействие с другими классами:**

**Пользователь**: описывает информационные данные и действия, связанные с пользователями, включая регистрацию, авторизацию, управление личным кабинетом и задачами. Подробное описание класса представлено в таблице 3.1.

**Таблица 3.1** – Описание класса Пользователь

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор пользователя. |
| Tasks | Задачи, связанные с пользователем. |
| email | Логин для создания и авторизации на сайте. |
| username | Пользовательское имя. |
| password | Пароль пользователя, используемый при входе в свой аккаунт. |

Окончание таблицы 3.1.

|  |  |
| --- | --- |
| role | Роль пользователя на сайте. Выбор ролей между USER и ADMIN. |

**Вакансия**: включает количество вакансий в сфере Java-разработки. Описание класса представлено в таблице 3.2.

**Таблица 3.2** – Описание класса Вакансия

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор вакансии. |
| Profession | Название вакансии. |
| NumberVacancies | Количество вакансий по данной профессии. |
| URL | Ссылка на запрос проводимой сайту поиска работы HH. |

**Требования:** Этот класс помогает пользователю ориентироваться в том, какие знания и навыки востребованы на рынке труда, в настоящее время. Описание класса представлено в таблице 3.3.

**Таблица 3.3** – Описание класса Требования

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор навыка. |
| SkillName | Название навыка. |
| SkillCount | Количество упоминаний. |

**Абстрактная задача:** описывает всю ключевые параметры, которые будут использоваться в классах наследниках. Описание класса представлено в таблице 3.4.

**Таблица 3.4** – Описание класса Абстрактная задача

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор задачи. |
| Title | Заголовок задачи. |
| Details | Подробное описание, что следует сделать. |
| Priorities | Выставление приоритета задачи. Выбор из 3 приоритетов: низкий, средний, высокий. |

Окончание таблицы 3.4.

|  |  |
| --- | --- |
| Completed | Поле отвечающие, завершена задача или нет. |
| CompletedAt | Дата завершения задачи. |

**Задача и подзадача**: обеспечивает возможность составления и управления задачами и подзадачами в планировщике. Это помогает пользователям планировать своё обучение, ставить краткосрочные цели и отслеживать их выполнение. Все поля являются наследниками класса Абстрактная задача.

**План**: содержит информацию о требуемых темах для изучения и рекомендации по их изучению. Описание класса представлено в таблице 3.5.

**Таблица 3.5** – Описание класса План

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор плана. |
| Developer | Название направления плана. |

**Темы**: включает перечень тем, которые пользователь должен изучить. Темы структурированы по сложности, что способствует более эффективной работе. Описание класса представлено в таблице 3.6.

**Таблица 3.6** – Описание класса Темы

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор темы. |
| Title | Заголовок темы. |
| Details | Описание для чего эта тема нужна и какие навыки должны быть приобретены. |
| Links | Ссылки на полезные источники для самостоятельного изучения. |

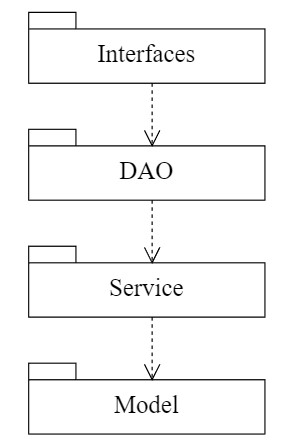
**Вопросы**: содержит список вопросов, которые часто задаются на собеседованиях по каждой теме. Это помогает пользователю не только изучить теоретический материал, но и подготовиться к практическим задачам и вопросам, которые могут встретиться на интервью. Описание класса представлено в таблице 3.7.

**Таблица 3.7** – Описание класса Вопросы

|  |  |
| --- | --- |
| Атрибут | Значение |
| Id | Идентификатор вопроса. |
| Questions | Вопрос, задаваемый по теме. |
| Answer | Ответ администратора. |
| Links | Ссылки на ответ на собеседовании из открытых источников. |
| NameTheme | Связь с классом темы и указание одной из имеющихся тем. |

## **3.5 Диаграмма пакетов**

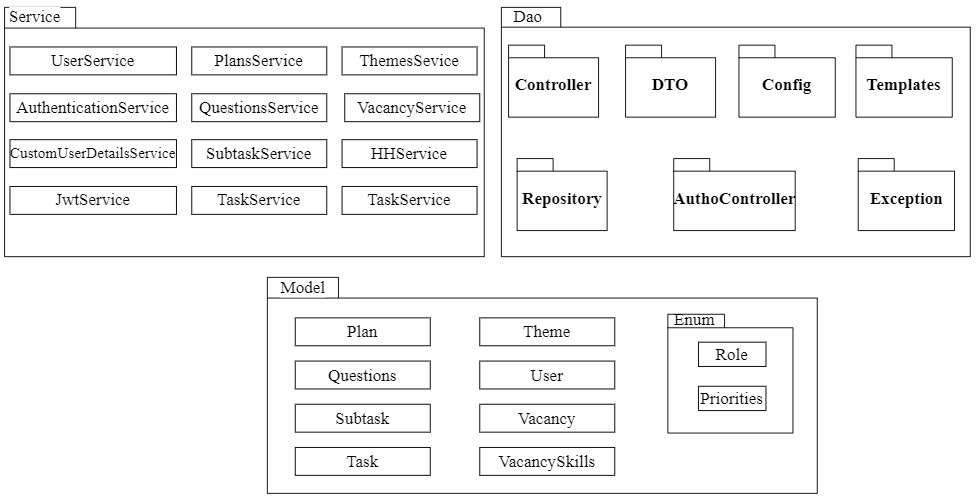
В данной главе будет рассмотрена диаграмма пакетов для веб-приложения, состоящего из четырёх основных пакетов: Interfaces, DAO, Service, и Domain. На рисунке 10 показана связь между этими пакетами.



*Рисунок 10 – Диаграмма пакетов для слоев*

* **Пакет Interfaces**: отвечает за взаимодействие с внешними пользователями и системами. Использует пакет DAO для работы с данными.
* **Пакет DAO (Data Access Object):** предоставляет доступ к данным и взаимодействует с базой данных, а также конфигурационные файлы. Зависит от пакета Service.
* **Пакет Service:** включает бизнес-логику приложения. Обрабатывает данные и реализует основные функциональные механизмы. Зависит от пакета Domain.
* **Пакет Domain:** содержит модели данных, которые определяют структуру объектов, используемых в приложении.

На следующем рисунке 11 представлен детальный вид на пакеты Service, DAO и Model.

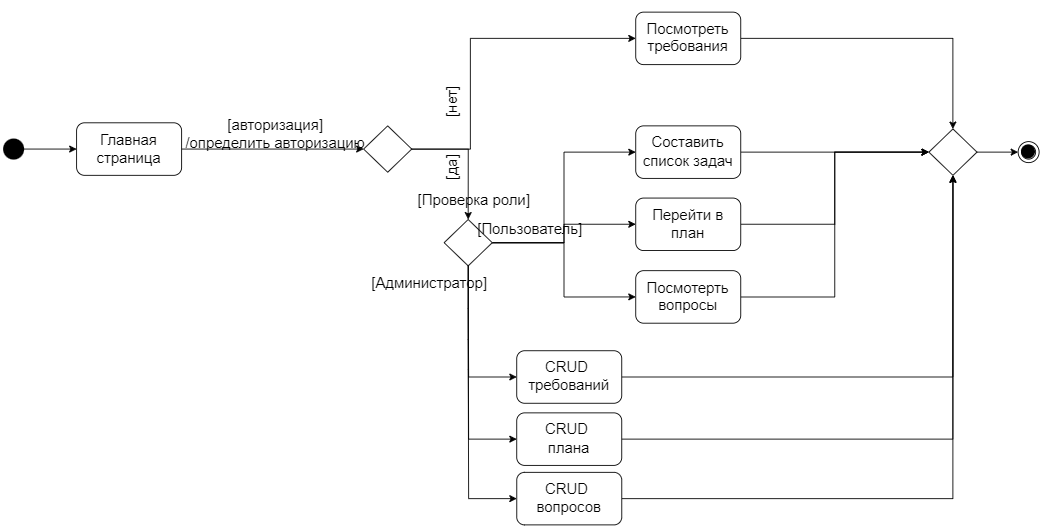


*Рисунок 11 – Диаграмма пакетов для слоев Dao, Service и Model*

Диаграмма пакетов демонстрирует четкую иерархию и разделение обязанностей между компонентами веб-приложения. Определение пакетов и их зависимостей позволяет обеспечить модульность и удобство поддержки приложения. Эффективная организация и управление зависимостями между различными пакетами способствует более четкому разделению ответственности и улучшению структурной целостности системы. Это особенно важно для веб-приложений, где модульность и структурированность играют ключевую роль в поддержке и расширении функциональности.

## **3.6 Диаграмма состояний**

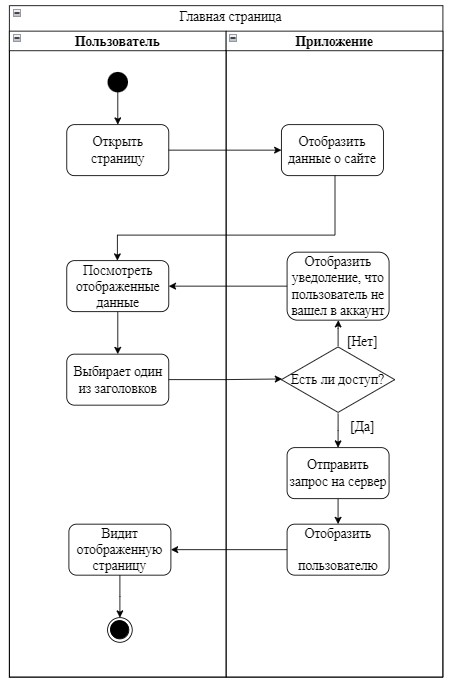
После определения иерархии компонентов и их модулей, важно также рассмотреть динамическое поведение системы. Диаграмма состояний предоставляет визуализацию возможных состояний объектов системы и переходов между этими состояниями в ответ на различные события. На рисунке 12 изображены допустимые состояния переходов пользователем из главной страницы.

*Рисунок 12 – Диаграмма пакетов для слоев Dao, Service и Model*

На рисунке представлены переходы в разные состояния из главной страницы, с помощью которого пользователь может использовать систему.

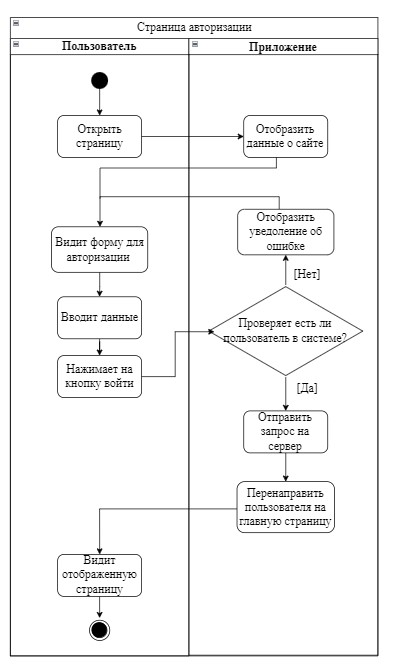
## **3.7 Диаграмма деятельности**

При открытии веб-приложения пользователю отображается главная страница с информацией о предназначении сайта и возможных направлениях перехода. На рисунке 13 представлена диаграмма деятельности для пользователя, вошедшего на веб-сайт.

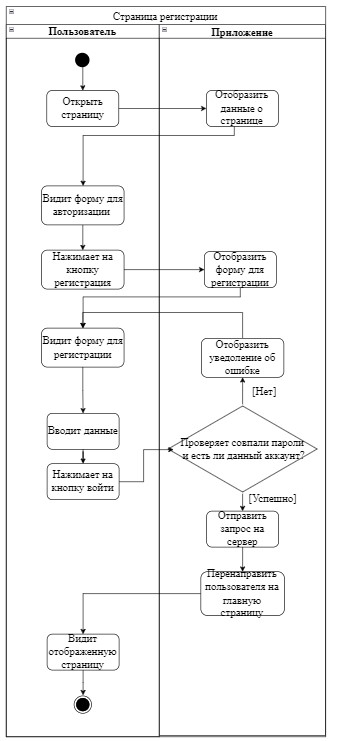


*Рисунок 13 – Диаграмма деятельности для «Главной страницы»*

На рисунке 14 и 15 изображена диаграмма деятельности, описывающая процессы авторизации и регистрации пользователей в системе.

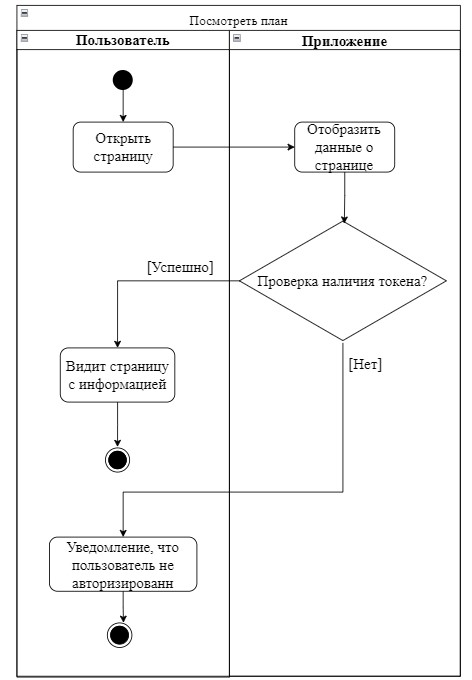


*Рисунок 14 – Диаграмма деятельности для «Страница авторизация»*



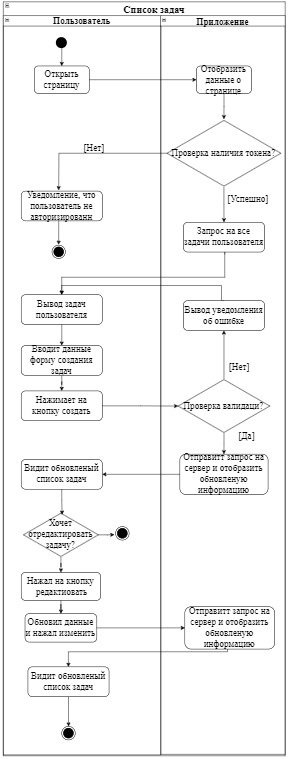
*Рисунок 15 – Диаграмма деятельности для «Страница регистрация»*

Далее, на рисунке 16 представлена диаграмма деятельности, отражающая функциональность "Посмотреть план". Здесь пользователь может ознакомиться с предлагаемыми рекомендациями по изучению тем.



*Рисунок 16 – Диаграмма деятельности для функциональности «Посмотреть план»*

На рисунке 17 рассмотрена диаграмма деятельности для функциональности "Список задач". Пользователь может видеть имеющиеся задачи и редактировать их после входа в аккаунт.



*Рисунок 17 – Диаграмма деятельности для функциональности «Список задач»*

**4. ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ**

## **4.1 Используемые технологии**

### 

### **4.1.1 Java**

Java – это объектно-ориентированный язык программирования, широко используемый для создания веб-приложений, мобильных приложений, а также корпоративных систем [2]. Благодаря своей кроссплатформенности и обширному набору библиотек, Java позволяет разработчикам создавать надежные и масштабируемые приложения. Одной из ключевых характеристик Java является принцип "напиши раз, запускай везде", позволяющий исполнять программы на любом устройстве с установленной JVM. В нашем проекте Java выступала основным языком программирования для разработки серверной части веб-приложения. Это обеспечило высокую степень гибкости и позволило интегрироваться с различными внешними системами и сервисами.

### **4.1.2 Spring Boot**

Spring Boot – это фреймворк для упрощенной разработки на платформе Spring, который значительно ускоряет процесс создания микросервисов и компонентов серверной части систем [3,8]. Основные преимущества Spring Boot включают автоматическую настройку и встраивание серверов приложений, что значительно ускоряет процесс разработки. Этот фреймворк предоставляет множество предустановок, которые позволяют разработчикам сразу приступить к реализации функционала, не тратя время на базовую конфигурацию. В нашем проекте Spring Boot использовался для создания RESTful веб-служб, что позволило быстро разрабатывать и тестировать серверные API. Благодаря Spring Boot нам удалось существенно сократить время на реализацию и обеспечить высокое качество кода.

**4.1.3 Hibernate**

Hibernate – это ORM (Object-Relational Mapping) фреймворк для Java, который позволяет разработчикам легко взаимодействовать с базами данных за счет трансформации данных между объектами Java и реляционными базами данных. Преобразование данных, выполняемое с помощью Hibernate, упрощает управление данными и повышает производительность разработчиков, устраняя необходимость писать сложные SQL-запросы. В нашем проекте Hibernate использовался для управления сущностями базы данных и взаимодействия с PostgreSQL. Это обеспечило единообразие кода и минимизировало вероятность ошибок при работе с данными, а также позволило сосредоточиться на бизнес-логике приложения.

**4.1.4 Maven**

Maven – это инструмент для управления проектами и автоматизации сборок, который предоставляет удобный способ управления зависимостями и выполнения различных фаз сборки проекта. Он поддерживает стандартизированные структуры проектов и предоставляет богатый набор плагинов для интеграции с другими инструментами. С Maven можно легко интегрировать проект с другими инструментами, такими как Docker и Jenkins, что улучшает автоматизацию разработки и сборки. В нашем проекте Maven использовался для управления зависимостями и автоматизации процесса сборки приложения. Это значительно упростило управление проектом и ускорило процесс разработки, обеспечив надежную и предсказуемую сборку.

**4.1.5 PostgreSQL**

PostgreSQL – это мощная объектно-реляционная система управления базами данных, поддерживающая сложные запросы и процедуры [10]. PostgreSQL отличается высокой производительностью, гибкостью и богатым набором функций. В нашем проекте PostgreSQL выступала в качестве основной базы данных для хранения данных приложения. Мы использовали PostgreSQL для обеспечения надежного и эффективного хранения данных, что позволило нам обеспечить высокую производительность.

**4.1.6 Swagger**

Swagger – это набор инструментов для разработки, развертывания и документации API[11]. Он предоставляет интерфейс для автоматической генерации документации API, что упрощает обмен данными и тестирование. В нашем проекте Swagger использовался для документирования RESTful API, созданных с использованием Spring Boot.

**4.1.7 Postman**

Postman – это популярный инструмент для тестирования API. Он предоставляет интуитивно понятный интерфейс для создания, тестирования и документирования RESTful API. В нашем проекте Postman использовался для тестирования и верификации API, разработанных с использованием Spring Boot.

**4.1.8 Thymeleaf**

Thymeleaf – это современный серверный шаблонный движок для Java. Он предназначен для создания динамических веб-страниц с поддержкой различных видов шаблонов (HTML, XML и другие). В нашем проекте Thymeleaf использовался в качестве шаблонизатора на стороне клиента, что позволило создавать интерактивные и динамические веб-страницы.

**4.1.9 JavaScript**

JavaScript – это язык программирования, используемый для создания интерактивного контента на веб-страницах. Он позволяет добавлять функциональность и динамичность к веб-приложениям. В нашем проекте JavaScript использовался для создания интерактивного интерфейса на веб-страницах, разработанных с использованием Thymeleaf.

## **4.2 Архитектура приложения**

### **4.2.1 Архитектурный паттерн Model-View-Controller (MVC)**

При разработке веб-приложения был использован архитектурный паттерн Model-View-Controller (MVC), который обеспечивает структурированное и удобное разделение приложения на три основные компоненты:

**Model** (Модель) – отвечает за управление данными и бизнес-логикой. Представляет собой логическую часть приложения, где происходит взаимодействие с базой данных, выполнение бизнес-правил и обработка данных.

**View** (Представление) – отвечает за отображение данных пользователю. В представлении осуществляется отображение данных, полученных от модели, в интерактивный интерфейс, воспринимаемый пользователем.

**Controller** (Контроллер) – осуществляет взаимодействие между моделью и представлением. Контроллер обрабатывает пользовательские запросы, инициирует изменение данных в модели и определяет, какое представление использовать для отображения данных.

### **4.2.2 Преимущества использования MVC**

Применение архитектурного шаблона MVC позволяет разделить логику работы веб-приложения на отдельные компоненты и упростить процесс разработки и поддержки. Ключевые преимущества применения MVC в нашем проекте включают:

1. Обеспечивает четкое разделение обязанностей между компонентами, что способствует проведению параллельной разработки и упрощает сопровождение приложения.
2. Благодаря изолированности логики, ее проще тестировать и отлаживать, а представление можно изменять без риска нарушения бизнес-логики.
3. Разработчики могут легко обновлять или заменять любой из компонентов без необходимости внесения изменений в остальные части системы.
4. Компоненты можно повторно использовать в разных частях приложения или даже в других проектах.

### **4.2.3 Принцип работы MVC в нашем приложении**

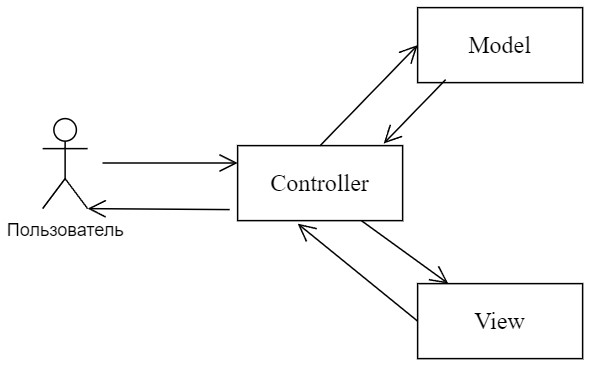
В проекте архитектурный паттерн MVC был реализован следующим образом:

**Модель**: в основе модели используется Hibernate для ORM (Object-Relational Mapping) взаимодействия с базой данных PostgreSQL, что позволило упростить управление базой данных и взаимодействие с ней.

**Представление**: для создания пользовательских интерфейсов используется Thymeleaf, который обеспечивает динамическое отображение данных, предоставляемых моделью.

**Контроллер**: при помощи Spring Boot были реализованы контроллеры, обрабатывающие HTTP-запросы, выполняющие бизнес-логику и определяющие, какие представления необходимо использовать для оформления ответа пользователю.

Схема взаимодействия компонент паттерна MVC в приложении представлена на рисунке 18. Данная схема иллюстрирует, как контроллеры взаимодействуют с моделью и представлением для обработки пользовательских запросов и генерации ответов.



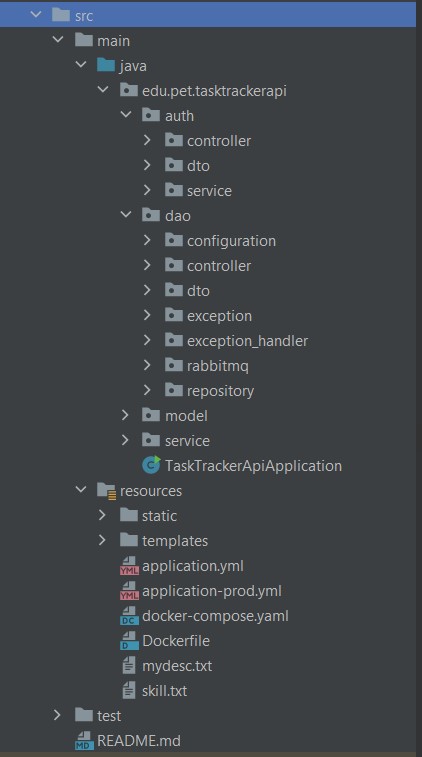
*Рисунок 18 – Схема взаимодействия MVC*

Архитектурный паттерн MVC значительно упростил процесс разработки веб-приложения, обеспечив структурированное разделение логики на независимые компоненты. Это позволило улучшить управляемость и сопровождаемость кода, снизить риск ошибок и повысить эффективность работы команды разработчиков.

**5. РЕАЛИЗАЦИЯ**

## **5.1 Структура проекта**

Разработка программного обеспечения осуществлялась с использованием языка программирования Java и фреймворка Spring Boot в интегрированной среде разработки IntelliJ IDEA. В процессе создания использовались принципы чистой архитектуры и архитектурный шаблон моделирования MVC (Model-View-Controller). Дерево каталогов приложения продемонстрировано на рисунке 19.



*Рисунок 19 – Структура папок приложения*

Как видно из рисунка, структура проекта разбита на следующие ключевые компоненты:

**Model** (Модель): Этот пакет содержит классы, представляющие данные приложения и бизнес-логику. Взаимодействие с базой данных осуществляется с помощью ORM фреймворка Hibernate.

**Service** (Сервисы): В пакете сервисов находятся классы, реализующие основную бизнес-логику приложения. Сервисы обеспечивают промежуточный уровень между контроллерами и моделью. Также в ресурсах у нас указаны пакеты templates и static, которые отвечают за генерацию страниц. И еще внешние файлы для подключения к внешним интерфейсам, в нашем случае только к БД.

**Dao** (Объекты доступа к данным): Пакет содержит интерфейсы и классы, которые отвечают за взаимодействие с клиентской частью и базой данных.

Помимо основных компонентов, структура проекта включает несколько дополнительных пакетов для обеспечения дополнительной функциональности и удобства разработки:

**auth**: Этот пакет выделен для обработки авторизации и регистрации пользователей. Он содержит контроллеры, сервисы и модели, необходимые для управления пользователями. Разделение пакета auth очень удобно для работы с пользователями, и он построен по тем же принципам, что и остальные пакеты в проекте.

**resources**: В этом пакете хранятся все необходимые для работы приложения ресурсы:

**templates**: Пакет включает в себя шаблоны, используемые Thymeleaf для генерации динамических веб-страниц.

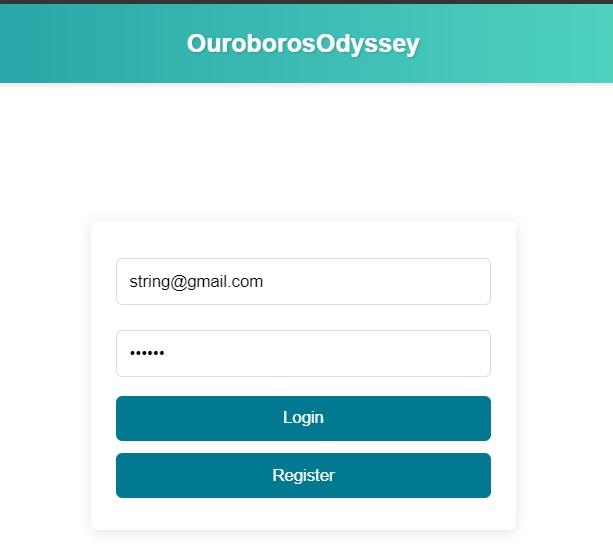
**static**: Пакет содержит статические ресурсы, такие как CSS-файлы, изображения и JavaScript-файлы.

**application**.**properties**: Конфигурационный файл, в котором указаны параметры для подключения к базе данных PostgreSQL и другие настройки приложения.

## **Реализация авторизации и регистрации пользователя**

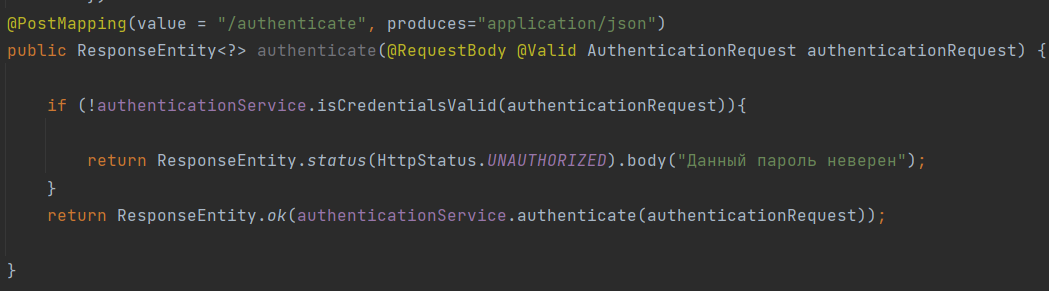
### **5.2.1 Авторизация пользователя**

Рассмотрим поэтапно реализацию функционала авторизации и регистрации пользователя в системе. На рисунке 20 представлена страница авторизации.



*Рисунок 20 – Форма авторизации*

На базовой форме присутствуют поля для ввода email и password для авторизации пользователя. После ввода данных и нажатия кнопки Login происходит отправка запроса на сервер, на рисунке 21.



*Рисунок 21 – Контроллер auth, запрос авторизации*

После отправки запроса проверяются введенные данные пользователя, как показано на Рисунке 22.

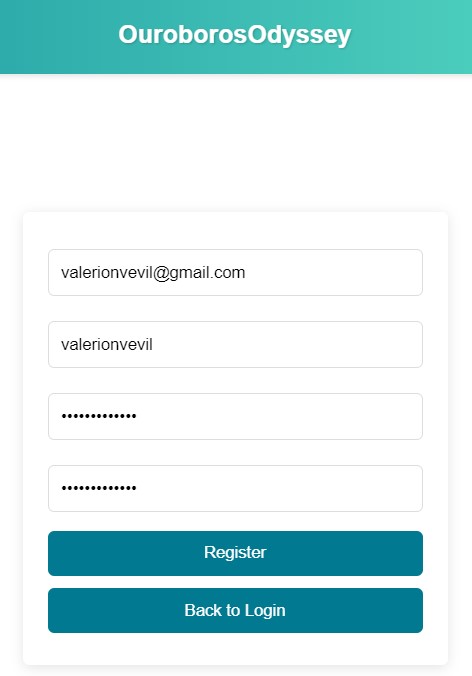


*Рисунок 22 – Проверка данных*

Если пользователь успешно проходит проверку, его перенаправляют на другую страницу.

### **5.2.2 Регистрация пользователя**

Если пользователь еще не зарегистрирован, то после нажатия на кнопку Register ему откроется форма для регистрации, на рисунке 23.



*Рисунок 23 – Форма регистрации*

После ввода всех данных и нажатия кнопки регистрации клиент отправит запрос на сервер, на рисунке 24. В случае успешного получения данных проводится проверка существования данного email в базе. Если он уже существует, сервер вернет статус 400. В противном случае вызывается метод регистрации.



Рисунок 24 – Контроллер auth, запрос регистрации

В методе регистрации, если почта имеет определённое названием, ей будет присвоена роль администратора, в остальных случаях пользователя, на рисунке 25.

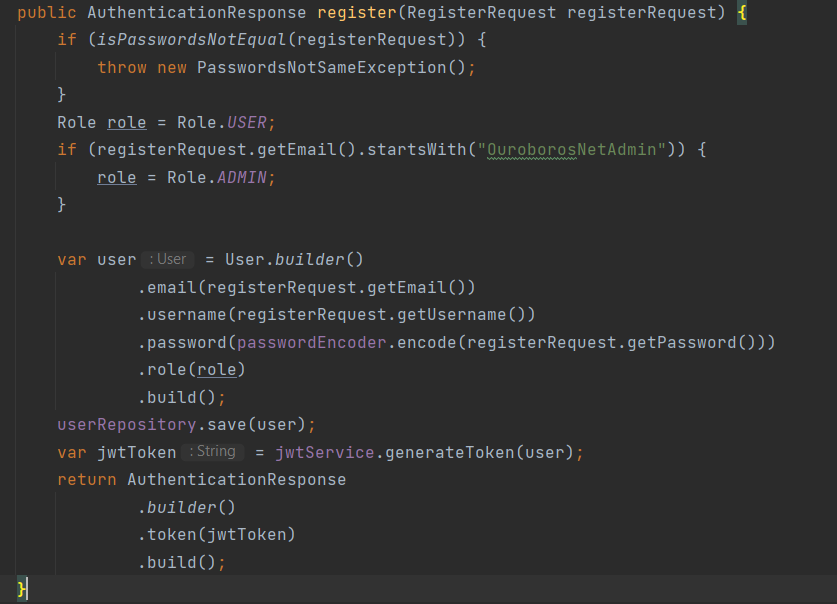


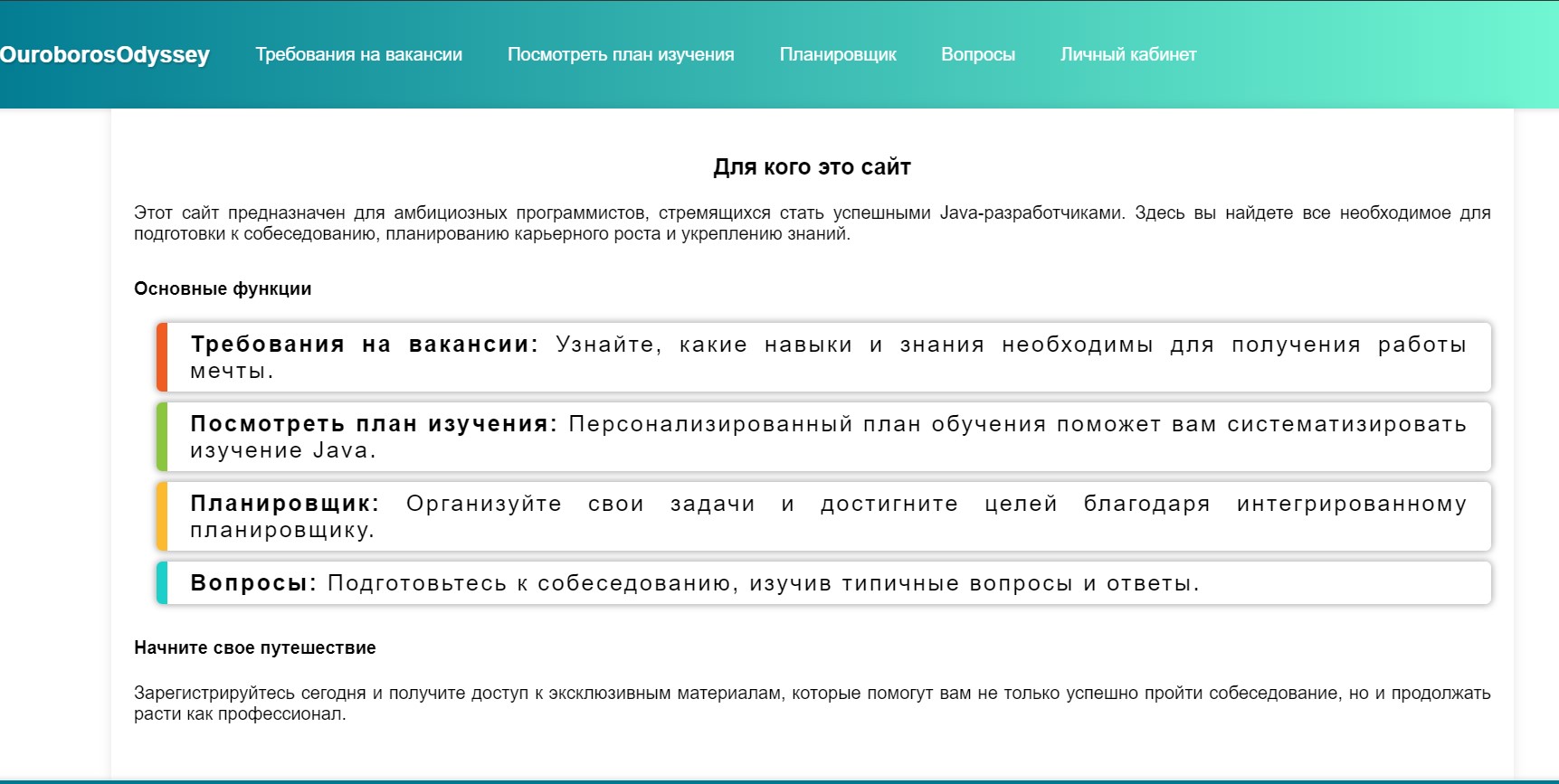
Рисунок 25 – Регистрация пользователя

После успешной регистрации или авторизации пользователю будет передан токен со сроком действия один час. По истечении этого срока потребуется повторная авторизация.

Таким образом, процесс авторизации и регистрации обеспечивает безопасность и управление доступом на основе токенов, что позволяет удобно и эффективно контролировать доступ пользователей к ресурсам веб-приложения.

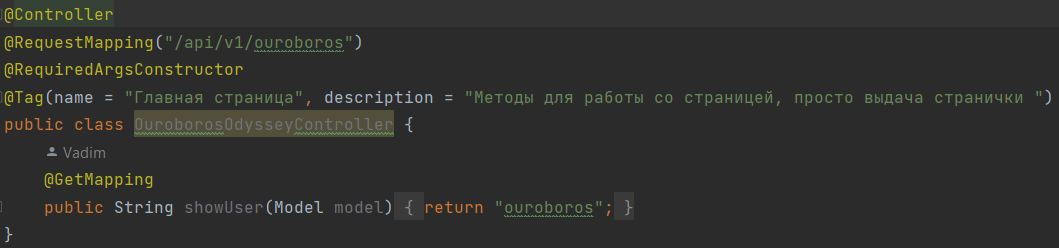
## **Реализация главной страницы**

После успешной авторизации\регистрации, пользователя перенаправит на главную страницу. Где будет коротко объяснено для чего служит этот сайт и будут даны варианты перехода других страниц. Как выглядит визуальная страница на рисунке 26.



*Рисунок 26 – Главная страница*

При переходе на главную станицу, вызовется метод get для отображения страницы пользователю.



*Рисунок 27 – Метод для отображения страницы*

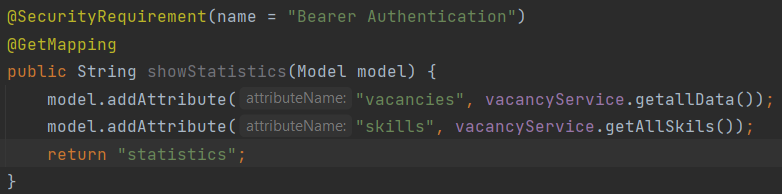
## **5.4 Реализация требований на вакансии**

При входе на страницу требований будут отображены данные о навыках, собранные из открытых должностей на странице HeadHunter, и выведены первые 100 наиболее частых. Это отобразится в виде таблицы и круговой диаграммы, рисунок 28.



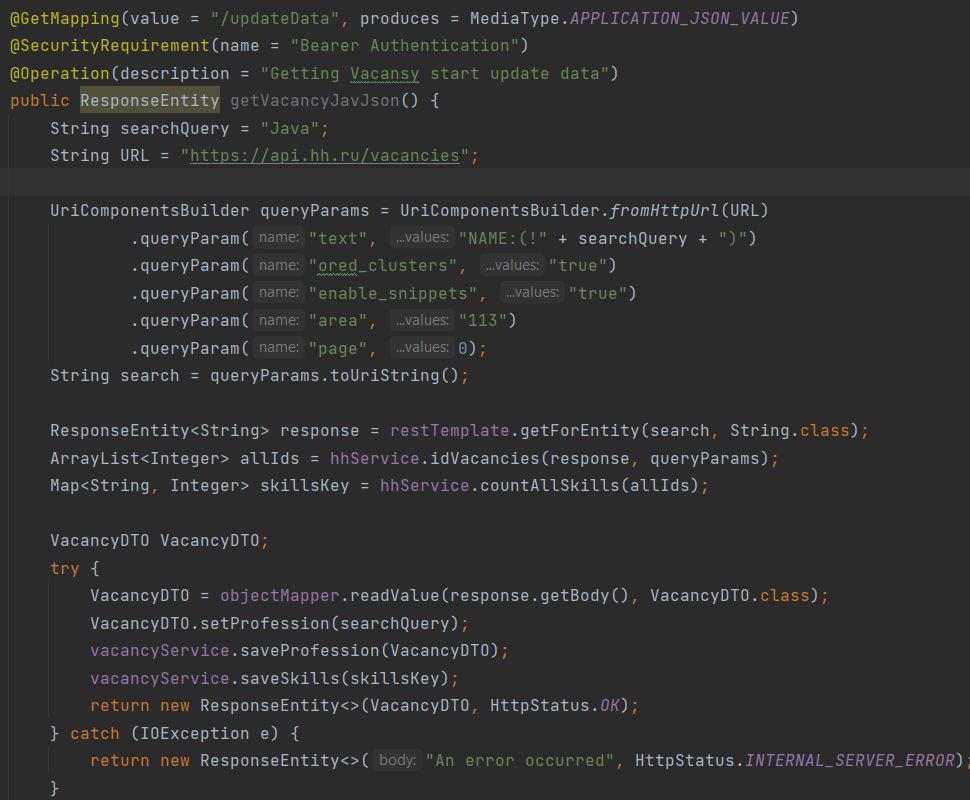
*Рисунок 28 – Страница требований*

Для вызова данных из базы данных и отображения их на странице используется метод, показанный на рисунке 29.



*Рисунок 29 – Запрос на получение данных из базы данных*

Чтобы обновить данные в нашей БД, используется метод, который проходит все вакансии, собирает количество навыков и сохраняет первые 100 в базе данных, на рисунке 30, метод для обновления данных.



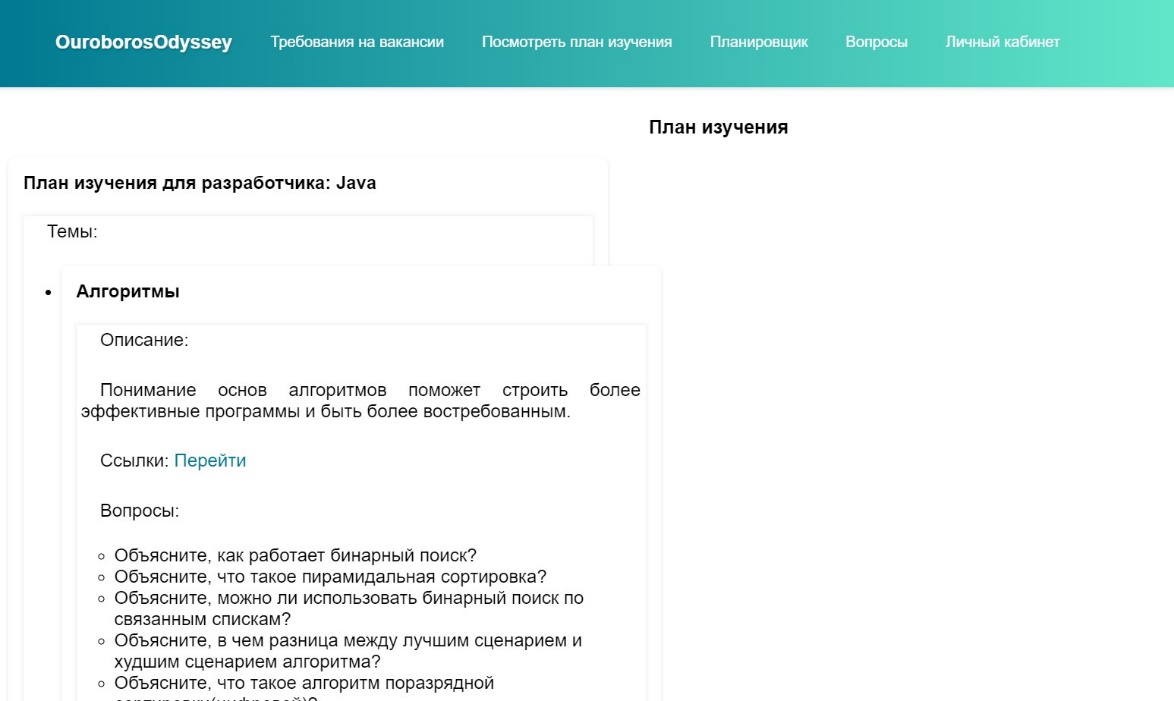
*Рисунок 30 – Метод для обновления данных о направлении Java*

В данной главе была разработана и реализована система для сбора и отображения данных о навыках, востребованных на должностях. Реализован автоматический сбор данных и их визуализация в виде таблицы и круговой диаграммы, а также возможность обновления данных.

## **5.5 Реализация плана изучения**

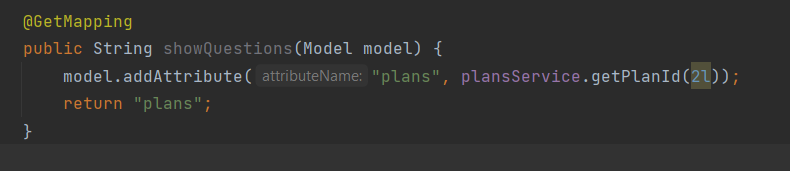
В процессе реализации функциональности просмотра плана изучения была разработана страница, представленная на рисунке 31.

При входе на страницу отправляется запрос на сервер, после чего пользователю отображается данная страница. На странице в верхней части выводится название плана по специальности (в нашем случае, Java). Ниже размещены названия тем и краткое описание к ним, а также ссылки на сторонние источники для углубленного изучения.



*Рисунок 31 – Страница плана изучения*

Вызываемый метод плана, на рисунке 32. Он выводит нашу станицу, по указанному id.



*Рисунок 32 – Метод вызова страницы*

В данной главе была успешно реализована функциональность, позволяющая пользователям просматривать план изучения. Она повышает удобство и эффективность использования образовательных ресурсов. Она предоставит пользователям четкую и структурированную информацию о темах, а также удобный доступ к дополнительным учебным материалам и вопросам для проверки. Такая интеграция способствует более глубокому и систематическому обучению, что особенно важно для специализированных областей, таких как Java.

## **5.6 Реализация вопросов на вакансии**

Разработанная страница с вопросами предоставляет пользователям возможность ознакомиться с часто встречающимися вопросами на собеседованиях и попытаться самостоятельно дать на них ответы. Вопросы удобно разбиты по темам, что позволяет пользователям сортировать их с использованием фильтра и изучать те вопросы, которые вызывают наибольшие затруднения.

На рисунке 33 представлена клиентская часть страницы, обеспечивающая удобное взаимодействие для пользователя.

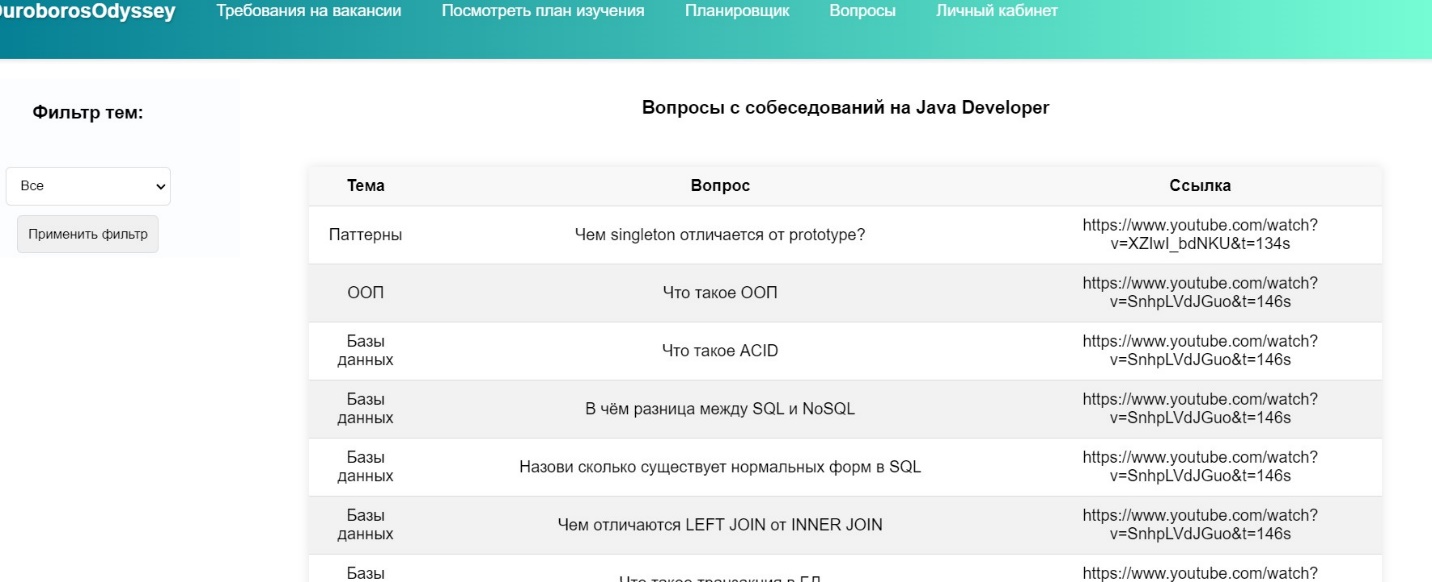
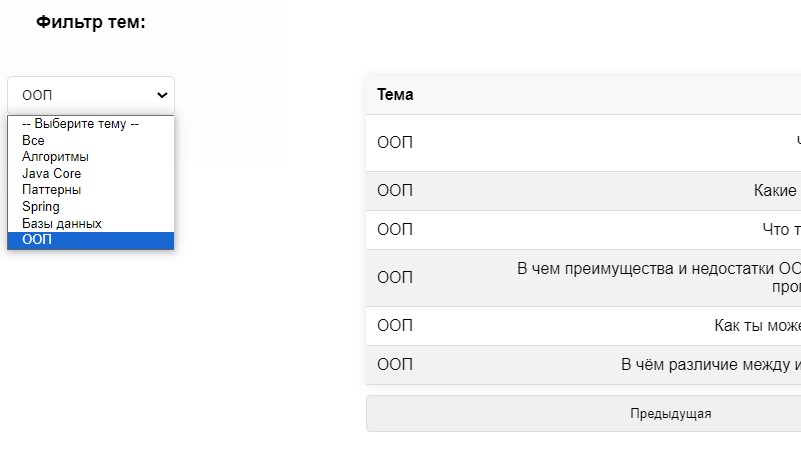


Рисунок 33 – Страница с отображением вопросов

Когда пользователь выбирает тему и нажимает на соответствующую кнопку, отправляется запрос на сервер, после чего страница обновляется перед пользователем, как показано на рисунке 34.



*Рисунок 34 – Возможность отфильтровать тему*

Разработанная страница с вопросами для подготовки к собеседованиям значительно улучшает образовательный процесс, предлагая удобную и структурированную платформу для самостоятельной практики.

## **5.7 Реализация планировщика**

Разработанная страница с реализацией планировщика предоставляет пользователям удобный интерфейс для управления своими задачами. При входе на страницу система проверяет наличие действительного токена у пользователя. Если токен еще не истек, пользователю предоставляется доступ к данной странице.

На странице планировщика отображаются списки задач пользователя с возможностью их редактирования и добавления новых. Это представлено на рисунке 35.

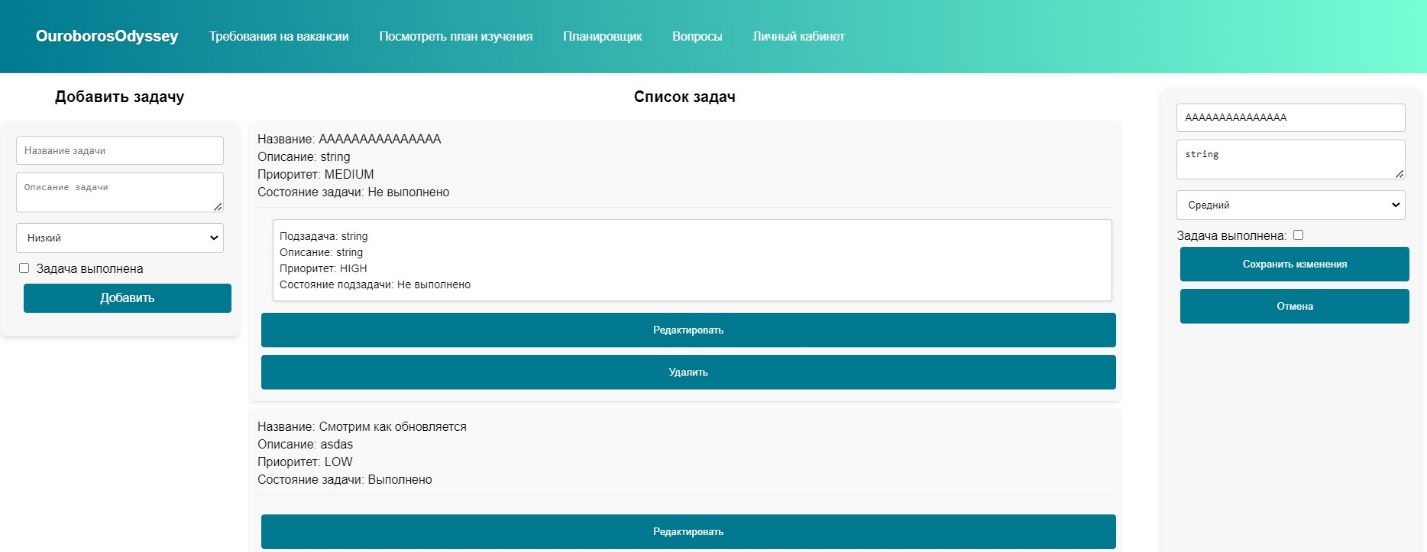


Рисунок 35 – Страница планировщика

При входе на страницу отправляется запрос от клиента к серверу, где после успешной проверки токена предоставляется доступ к функциональности планировщика. В списке задач пользователь может:

Просматривать текущие задачи.

Выбирать одно из двух действий: удалить задачу или отредактировать ее.

Для удаления задачи отправляется её id на сервер, и после успешного удаления изменения отображаются перед пользователем. На рисунке 36 представлен метод контроллера на сервере, а на рисунке 37 – запрос, который отправляется со стороны клиента.

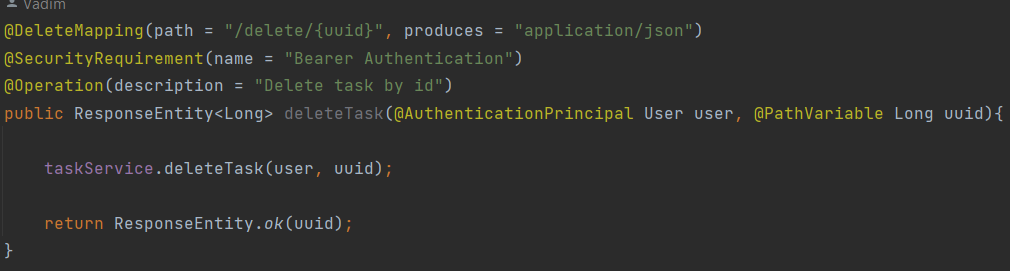


Рисунок 36 – Запрос со стороны сервера

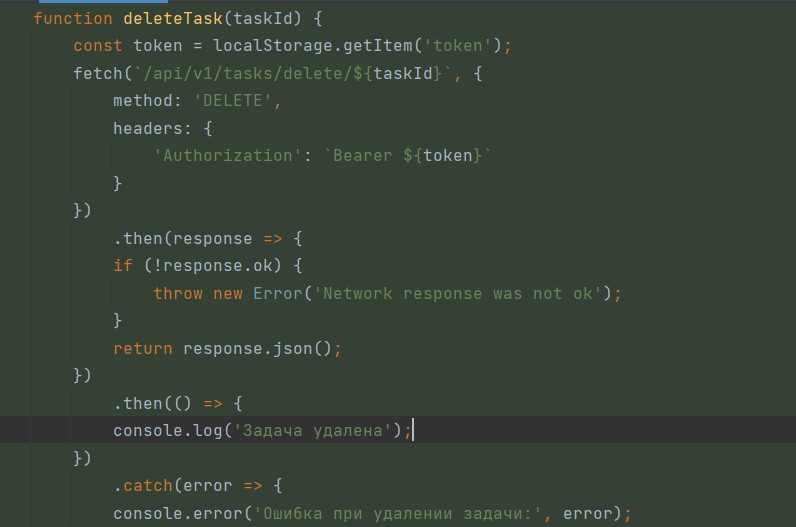
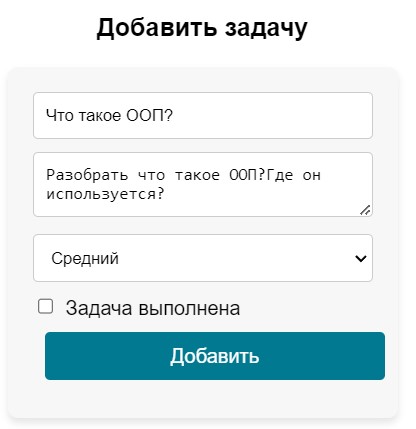


Рисунок 37 – Запрос со стороны клиента

Если перед пользователем будет стоять задача создать, новую задачу, он может обратиться к форме и добавить себе задачи, это показано на рисунке 38.



*Рисунок 38 – Функционал добавить задачу*

Реализация страницы планировщика значительно повышает удобство управления задачами для пользователей, предоставляя им четкий и функциональный интерфейс для работы. Интеграция возможностей редактирования и удаления задач делает планировщик гибким инструментом для повседневного использования.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы было разработано веб-приложение для планирования задач и подготовки к собеседованию по направлению Java. Созданное приложение предоставляет пользователю следующие функции:

* Добавление и редактирование задач.
* Маркировка задач.
* Анализ требований по Java.
* База данных вопросов и ссылок на собеседования.

Также в ходе выполнения выпускной работы, были выполнены следующие задачи:

Во-первых, был проведён анализ предметной области планирования задач и подготовки к техническому собеседованию. В рамках этой задачи изучены следующие аспекты:

* Методы подготовки (обучения).
* Подходы и методы эффективного планирования задач.

Во-вторых, выполнен обзор существующих веб-приложений с аналогичным функционалом и оценена их простота использования.

В-третьих, определены требования к функционалу разрабатываемого приложения.

В-четвёртых, изучены используемые инструменты разработки, которые подходят для реализации требуемого функционала.

В-пятых, осуществлена реализация требуемого функционала путем проектирования архитектуры веб-приложения, включая:

* Составление описаний вариантов использования.
* Создана схема взаимодействия.
* Построение диаграммы пакетов, состояний и деятельности.
* Формирование моделей предметной области.

В-шестых, разработано само веб-приложение.

Таким образом, разработанное веб-приложение представляет собой эффективный инструмент для планирования задач и подготовки к техническому собеседованию по Java, обеспечивая пользователям необходимые функции для успешного освоения материалов и подготовки к интервью. Полученные результаты могут быть использованы для дальнейшего совершенствования методологий планирования задач и подготовки к собеседованиям, а также для разработки аналогичных приложений в других областях.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Фаулер M. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с., ил. ISBN 5-93286-060-Х URL: https://www.litres.ru/book/martin-fauler/uml-osnovy-3-e-izdanie-24500318/ (дата обращения 10.12.2023)
2. Java Documentation [Электронный ресурс] URL: <https://docs.oracle.com/en/java/> (дата обращения 12.12.2023)
3. Spring Framework Documentation [Электронный ресурс] URL: <https://docs.spring.io/spring-framework/reference/> (дата обращения 12.12.2023)
4. Классификация методов обучения [Электронный ресурс] URL: <https://paidagogos.com/klassifikatsiya-metodov-obucheniya.html> (дата обращения 12.01.2024)
5. Классификация методов обучения (Ю.К. Бабанский) [Электронный ресурс] URL: <https://ped.bobrodobro.ru/54044> (дата обращения 12.01.2024)
6. Моисеев А. Н. Основы языка UML: учебное пособие / А. Н. Моисеев, М. И. Литовченко; Нац. исслед. Томский гос. ун-т. - Томск: Издательство Томского государственного университета, 2023. URL: <http://vital.lib.tsu.ru/vital/access/manager/Repository/koha:001006819>
7. Уоллс К.: Spring в действии. 6-е изд./ пер. с англ. А. Н. Киселева. – М.: ДМКПресс, 2022. – 544 с.: ил. – (Библиотека программиста) (дата обращения 02.03.2024)
8. Bauer C., King G. Java Persistence with Hibernate. Second Edition. – Greenwich: Manning Publications, 2015. – 608 p. URL: https://www.manning.com/books/java-persistence-with-hibernate-second-edition (дата обращения 05.05.2024).
9. PostgreSQL Documentation [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения 04.05.2024).
10. Swagger Documentation [Электронный ресурс]. URL: https://swagger.io/docs/ (дата обращения 15.05.2024).