Титульный лист

Лист задания

Реферат

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Реферат

Лит.

Листов

1

*74218051, 2023*

У

Николайчук А.Н.

Пояснительная записка дипломного проекта содержит 79 страниц пояснительной записки, 8 таблиц, 8 формул, 66 иллюстраций, 17 источников литературы, 8 приложений.

Веб-приложение, REACT, JAVASCRIPT, AXIOS, REACT BOOTSTRAP, java, Spring framework, mysql, DOCKER.

Целью дипломного проекта является разработка веб-приложения для управления IT-проектами.

В первой главе проводится аналитический обзор литературы по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, и показывается поведения системы при разных внештатных ситуациях.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе приводится расчет экономических параметров и себестоимости программного средства.

Объем графической части дипломного проекта составляет 1.75 листа А1.

Abstract

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Abstract

Лит.

Листов

1

*74218051, 2023*

У

Николайчук А.Н.

Explanatory note of the diploma project 79 pages of explanatory note, 8 tables, 8 formulas, 66 illustrations, 17 sources of literature, 8 appendices.

WEB APPLICATION, REACT, JAVASCRIPT, AXIOS, REACT BOOTSTRAP, java, Spring framework, mysql, DOCKER.

The aim of the diploma project is to develop a web application for IT-projects management.

The first chapter provides an analytical review of the literature on the subject of the diploma project.

The second chapter is devoted to a review of development tools and describes the technologies used during the project.

The third chapter describes the development process, the principles of functioning and the purpose of the created project components.

The fourth chapter describes a test case, with testing, as well as a demonstration of the behavior of the system in various emergency situations.

The fifth chapter describes a user manual that allows you to understand in detail the interface of the software tool.

The sixth chapter provides the calculation of economic parameters and the cost of software developed in the framework of the graduation project.

The volume of the graphic part of the diploma project is 1.75 sheets A1.

**Содержание**

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Содержание

Лит.

Листов

2

*74218051, 2023*

У

Николайчук А.Н.

[Введение 7](#_Toc136987313)

[1 Постановка задачи и обзор аналогичных решений 8](#_Toc136987314)

[1.1 Постановка задачи 8](#_Toc136987315)

[1.2 Обзор аналогов 9](#_Toc136987316)

[1.2.1 Приложение «Trello» 9](#_Toc136987317)

[1.2.2 Приложение «Notion» 10](#_Toc136987318)

[1.2.3 Приложение «Jira» 11](#_Toc136987319)

[1.3 Патентный поиск 12](#_Toc136987320)

[1.4 Выводы по разделу 13](#_Toc136987321)

[2 Проектирование программного средства 14](#_Toc136987322)

[2.1 Диаграмма вариантов использования 14](#_Toc136987323)

[2.2 Выбор средств реализации 15](#_Toc136987324)

[2.2.1 Основные языки программирования 15](#_Toc136987325)

[2.2.2 Фреймворки и библиотеки 16](#_Toc136987326)

[2.2.3 Система управления базами данных 17](#_Toc136987327)

[2.2.4 Вспомогательные инструменты 18](#_Toc136987328)

[2.3 Авторизация пользователей 19](#_Toc136987329)

[2.4 Контейнеризация 20](#_Toc136987330)

[2.5 Проектирование базы данных 22](#_Toc136987331)

[2.5.1 Таблица roles 23](#_Toc136987332)

[2.5.2 Таблица users 23](#_Toc136987333)

[2.5.3 Таблица employees 23](#_Toc136987334)

[2.5.4 Таблица tickets 24](#_Toc136987335)

[2.5.5 Таблица projects\_has\_employees 24](#_Toc136987336)

[2.5.6 Таблица comments 25](#_Toc136987337)

[2.5.7 Таблица logged\_time 25](#_Toc136987338)

[2.5.8 Таблица orders 26](#_Toc136987339)

[2.6 Проектирование основных алгоритмов 26](#_Toc136987340)

[2.6.1 Алгоритм перемещения задачи на «дэшборде» 26](#_Toc136987341)

[2.6.2 Алгоритм изменения прикрепленного к задаче файла 27](#_Toc136987342)

[2.6.3 Алгоритм авторизации заказчика в приложении 29](#_Toc136987343)

[2.6.4 Алгоритм изменения пароля от аккаунта 30](#_Toc136987344)

[2.7 Выводы по разделу 31](#_Toc136987345)

[3 Реализация программного средства 32](#_Toc136987346)

[3.1 Общая архитектура приложения 32](#_Toc136987347)

[3.2 Разработка серверной части 33](#_Toc136987348)

[3.2.1 Установка необходимых компонентов 33](#_Toc136987349)

[3.2.2 Выбор архитектуры 34](#_Toc136987350)

[3.2.3 Разработка контроллеров 37](#_Toc136987351)

[3.2.4 Разработка сервисов 38](#_Toc136987352)

[3.2.5 Разработка репозиториев 38](#_Toc136987353)

[3.3 Разработка клиентской части 39](#_Toc136987354)

[3.3.1 Установка необходимых компонентов 39](#_Toc136987355)

[3.3.2 Выбор архитектуры проекта 40](#_Toc136987356)

[3.3.3 Маршрутизация в приложении 41](#_Toc136987357)

[3.3.4 Валидация данных 42](#_Toc136987358)

[3.4 Выводы по разделу 43](#_Toc136987359)

[4 Тестирование программного средства 44](#_Toc136987360)

[4.1 Тестирование серверной части 44](#_Toc136987361)

[4.2 Тестирование клиентской части 49](#_Toc136987362)

[4.3 Выводы по разделу 54](#_Toc136987363)

[5 Руководство пользователя 55](#_Toc136987364)

[5.1 Развертывание приложения в среде Docker 55](#_Toc136987365)

[5.2 Руководство для сотрудника 57](#_Toc136987366)

[5.3 Руководство для менеджера 64](#_Toc136987367)

[5.4 Руководство для администратора 65](#_Toc136987368)

[5.5 Руководство для клиента 67](#_Toc136987369)

[5.6 Выводы по разделу 68](#_Toc136987370)

[6 Технико-экономическое обоснование проекта 69](#_Toc136987371)

[6.1 Общая характеристика разрабатываемого программного средства 69](#_Toc136987372)

[6.2 Исходные данные для проведения расчетов и маркетинговый анализ 70](#_Toc136987373)

[6.3 Обоснование цены программного средства 71](#_Toc136987374)

[6.3.1 Расчет затрат рабочего времени на разработку программного средства 71](#_Toc136987375)

[6.3.2 Расчет основной заработной платы 72](#_Toc136987376)

[6.3.3 Расчет дополнительной заработной платы 72](#_Toc136987377)

[6.3.4 Расчет отчислений в Фонд социальной защиты населения и по страхованию 73](#_Toc136987378)

[6.3.5 Расчет суммы прочих прямых затрат 73](#_Toc136987379)

[6.3.6 Расчет суммы накладных расходов 74](#_Toc136987380)

[6.3.7 Сумма расходов на разработку программного средства 74](#_Toc136987381)

[6.3.8 Определение цены, оценка эффективности 75](#_Toc136987382)

[6.4 Выводы по разделу 76](#_Toc136987383)

[Заключение 77](#_Toc136987384)

[Список использованных источников 78](#_Toc136987385)

[ПРИЛОЖЕНИЕ А Диаграмма вариантов использования 80](#_Toc136987386)

[ПРИЛОЖЕНИЕ Б Листинг классов 81](#_Toc136987388)

[ПРИЛОЖЕНИЕ В Логическая схема базы данных 88](#_Toc136987390)

[ПРИЛОЖЕНИЕ Г Блок-схема алгоритма перемещения задачи на «дэшборде» 89](#_Toc136987392)

[ПРИЛОЖЕНИЕ Д Блок-схема алгоритма изменения прикрепленного к задаче файла 90](#_Toc136987394)

[ПРИЛОЖЕНИЕ Е Блок-схема алгоритма авторизации заказчика в приложении 91](#_Toc136987396)

[ПРИЛОЖЕНИЕ Ж Блок-схема алгоритма изменения пароля от аккаунта 92](#_Toc136987398)

[ПРИЛОЖЕНИЕ И Структурная схема архитектуры приложения 93](#_Toc136987400)

[ПРИЛОЖЕНИЕ К Листинг классов SecurityConfig и JwtFilter 94](#_Toc136987402)

Введение

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Введение

Лит.

Листов

1

*74218051, 2023*

У

Николайчук А.Н.

В настоящее время практически все сферы деятельности используют в своей работе информационные технологии, а сама IT-сфера тем более не является исключением. Создание IT-проекта – это очень многозадачный процесс: в него вовлечены десятки людей, тратятся месяцы времени. Поэтому для разработки IT-проектов были созданы различные методы управления проектами. Это помогает планировать время, экономить ресурсы, оптимизировать процесс разработки.

Чтобы сделать свой проект более управляемым, логично разбить все процессы на последовательные этапы. Именно на такой линейной структуре базируется традиционное проектное управление. В настоящее время все большую популярность приобретают так называемые Agile-методологии, в особенности модель разработки программного обеспечения Kanban. Ее классическая реализация – это таблица с тремя столбцами «Нужно сделать», «В работе» и «Сделано». Но столбцов может быть и больше, например, по количеству этапов в проекте или на производстве, а их названия также могут меняться по усмотрению компании. Все дела по умолчанию заносятся в первый столбец отдельной строкой или на отдельном стикере, карточке. Потом они перетаскиваются из одного в другой, пока, не попадут в последний столбец. Благодаря такому «перетаскиванию» можно легко отследить прогресс команды.

Целью проектирования дипломного проекта является разработка веб-приложения, реализующего методологию Kanban. Это приложение для управления IT-проектами на основе вышеупомянутой методологии, где сотрудники смогут создавать проекты, задачи, редактировать их, менять их очередность на «дэшборде».

Для достижения цели необходимо соблюсти последовательность действий:

* спроектировать и разработать структуру базы данных;
* спроектировать пользовательский интерфейс клиента;
* распределить возможности каждой из ролей в приложении;
* разработать серверную часть проекта;
* разработать клиентскую часть проекта;
* провести тестирование проекта.

Веб-приложение разрабатывается с использованием языков программирования Java и JavaScript, библиотеки React, фреймворка Spring.

1. Постановка задачи и обзор аналогичных решений

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 01.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

1 Постановка задачи и обзор аналогичных решений

Лит.

Листов

6

*74218051, 2023*

У

Николайчук А.Н.

Для создания программного средства сначала необходимо изучить существующие аналоги и выделить их основные недостатки и преимущества, определить ключевые особенности приложений в данном направлении. Затем на основе проведенного анализа необходимо определить задачи при разработке программы. Также должен быть осуществлен поиск по существующем патентам для того, чтобы выяснить, существует ли уже на рынке реализация подобного функционала. Все вышеописанные действия были выполнены в данной главе.

* 1. Постановка задачи

Следуя тому, что задача работы звучит как веб-приложение для управления IT-проектами, прежде всего, должно быть разработано веб-приложение, которое представляется пользователю как набор страниц с поддержкой маршрутизации, возможностью просматривать их в браузере, между которыми присутствует возможность перехода. Это позволит взаимодействовать с приложением при помощи как стационарных, так и мобильных операционных систем. Поскольку данное приложение относится к типу веб, следовательно, к нему можно осуществить доступ как из локальной сети, так и сети Интернет в целом. Веб-приложение должно взаимодействовать с централизованной базой данных, обеспечивающей хранение пользовательской информации. Само взаимодействие между клиентской и серверной частью должно базироваться на клиент-серверной архитектуре, связь и передача данных между ними должна осуществляться с использованием протокола HTTP.

Целью программного средства, реализуемого в дипломном проекте, является возможность создания и управления процессом разработки проектов, а также возможность управления командой, работающей над данным проектом. Помимо этого, приложение должно позволять просматривать статистику проделанной работы.

Проект должен выполнять следующие задачи:

* сохранение рабочей информации в централизованной базе данных;
* создание пользователей с заданными ролями;
* создание новых проектов;
* создание задач и распределение их по исполнителям;
* комментирование задач;
* просмотр статистики выполнения задач проекта.

Также проект должен обладать определенными бизнес-целями. Одной из целей дипломной работы является расчет и получение прибыли с продажи проекта. Таким образом, проект в дальнейшем можно развивать и предоставлять для продажи.

* 1. Обзор аналогов

Обзор аналогов является важным шагом перед началом проектирования какого-либо приложения. Данный этап позволяет выявить возможности, которые разрабатываемое приложение должно реализовывать в обязательном порядке, а также дополнительные функции, которые были бы отличительной особенностью приложения, и негативные стороны, которые лучше не допускать в приложении.

Обзор аналогов будет проводится на основе следующих критериев:

* доступность пользователям;
* наличие необходимого функционала для работы;
* наполненность профиля пользователей;
* осуществление поиска среди пользователей;
* наличие метаданных о проекте;
* наличие комментариев к задачам;
* наличие статистики о проделанной работе.

В качестве аналогов были рассмотрены приложения «Trello», «Notion», «Jira».

* + 1. Приложение «Trello»

Приложение «Trello» является, пожалуй, самым известным среди приложений для управления проектами и процессами с использованием Kanban-доски [1]. На рисунке 1.1 представлен ее пример в данном веб-приложении.

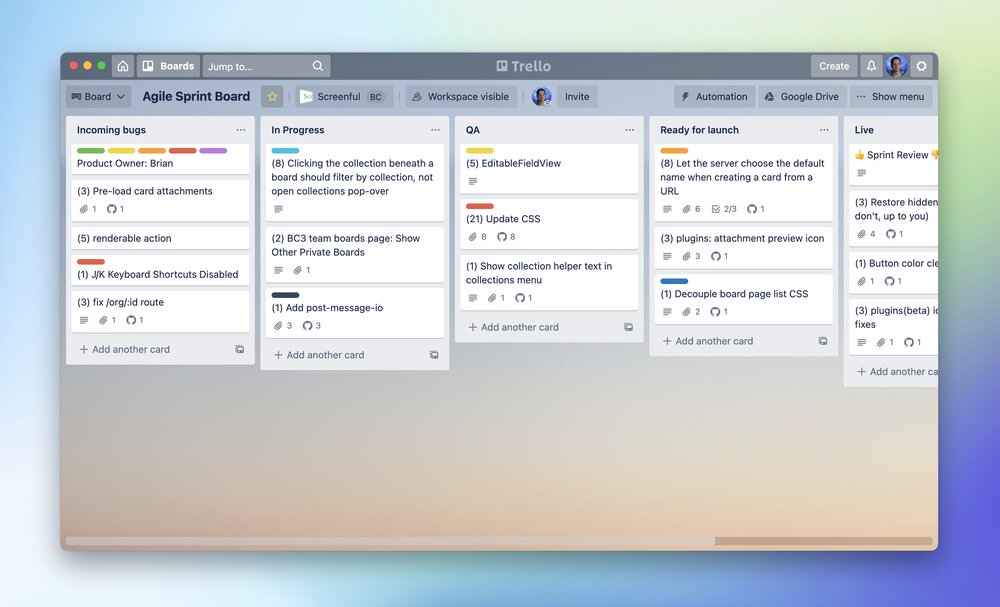


Рисунок 1.1 – Интерфейс приложения «Trello»

Важный плюс этого приложения – это доступность, любой желающий может воспользоваться данным приложением. Оно имеет бесплатную версию, обладает приятным дизайном, удобно в использовании. Для каждого проекта пользователи могут создавать отдельную доску, причем приватную или общедоступную, и внутри нее создавать, редактировать, распределять задачи. Внутри задач можно создавать чек-листы, то есть как бы подпункты задачи, а сами задачи можно комментировать. К задачам также можно прикреплять вложения, добавлять метки и другие метаданные. Присутствует переключатель между досками, можно произвести фильтрацию задач на доске, просмотреть участников доски. В приложении также есть возможность подключения уведомлений к конкретным задачам.

В основном данным приложением пользуются небольшие команды, оно отлично подходит под их потребности, а вот в случае с крупными проектами, это будет не самым лучшим выбором. В «Trello» отсутствует возможность просмотра какой-либо статистики. Также профили пользователей практически не содержат никакой информации о них: представлены только имя, никнейм и поле о себе. Более того к проектам нельзя добавить описания или какие-либо другие данные, из метаданных в проекте можно указать только имя.

В целом приложение обладает всем основным необходимым функционалом, но все же не является идеальным, поскольку нуждается в доработке определенных моментов, в том числе вышеперечисленных. «Trello» подойдет для любых целей, даже личных, например, для использования в качестве заметок.

* + 1. Приложение «Notion»

Веб-приложение «Notion» является еще одним из аналогов разрабатываемого проекта [2]. Данное приложение предоставляет возможность использования Kanban-доски, пример интерфейса которой представлен на рисунке 1.2.

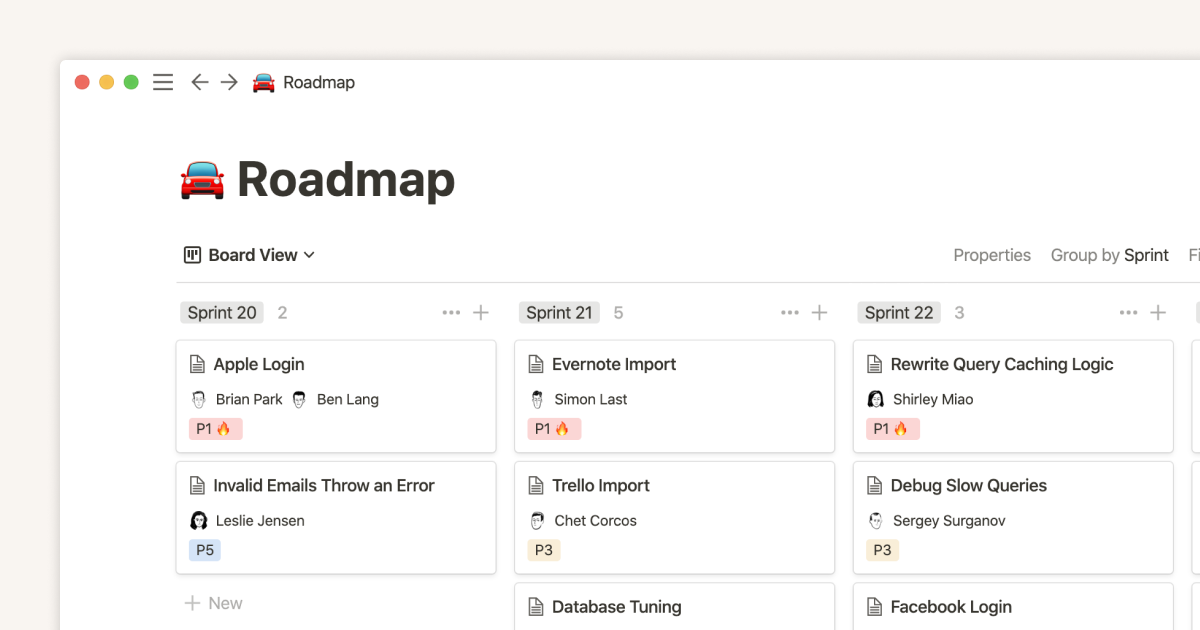


Рисунок 1.2 – Интерфейс приложения «Notion»

Отличительная особенность данного приложения и его преимущество – это возможность создания различных представлений для задач. Основным и главным видом является доска, также к ней можно добавить представление задач в виде таблицы, списка. Задачи можно группировать по критерию, фильтровать и сохранять отфильтрованные представления, что очень удобно для последующей навигации, присутствует поиск по задачам. В задачах можно прикреплять файлы, создавать списки, оставлять комментарии, помечать результаты после выполнения задачи. Приложение имеет приятный интерфейс, где нет ничего лишнего и который можно изменить соответствии со своими предпочтениями, и бесплатную версию.

Недостатки у приложения, к сожалению, также присутствуют. У пользователей нет полноценных профилей в приложении, невозможно осуществлять поиск среди пользователей, нет возможности добавить описание или другую информацию к проекту, нет возможности просмотра статистики по проекту и задачам.

Приложение «Notion» будет прекрасным помощником в организации некрупных процессов, поскольку обладает всеми необходимыми для этого базовыми функциями. Особенно удобно оно будет для использования в личных и даже бытовых целях. Однако приложение неприменимо для использования в качестве единого корпоративного менеджера проектов внутри компании, так как его функционал для этого слишком узок. Тем не менее, оно обладает большим потенциалом.

* + 1. Приложение «Jira»

Приложение «Jira» – это программный инструмент для управления проектами [3]. Оно очень популярно среди IT-компаний, поскольку обладает возможностями для менеджмента крупных проектов. «Jira» имеет бесплатную версию, однако она содержит ограничения по количеству человек в команде, а также значительные ограничения по дополнительному функционалу. Пример интерфейса данного приложения представлен ниже на рисунке 1.3.

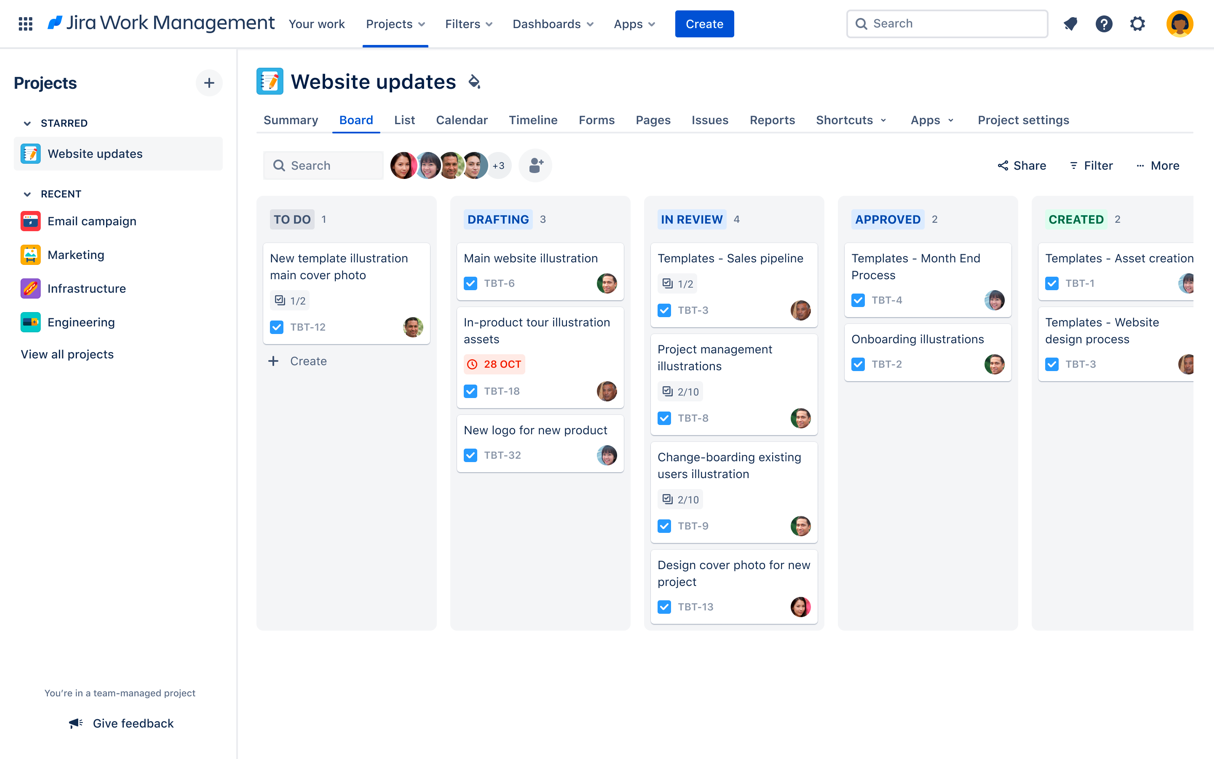


Рисунок 1.3 – Интерфейс приложения «Jira»

Данное приложение, как и другие аналоги, позволяет создавать проекты, задачи к ним, представлять задачи в виде доски, выполнять поиск и фильтрацию по задачам, комментировать задачи, прикреплять файлы к задачам. Помимо этого, в отличие от ранее рассмотренных приложений, «Jira» позволяет добавлять описания к проектам, просматривать статистику. Важным достоинством приложения является возможность его дополнения персональными улучшениями, интегрирования с другими сервисами, что позволяет настроить его полностью под нужды компании. Помимо представления задач в виде «дэшборда» «Jira» также позволяет создавать диаграмму Ганта, матрицу идей, планировщики спринтов и многое другое.

Все же в приложении не предусмотрены некоторые детали. Профили пользователей содержат минимум информации: никнейм, имя, электронная почта. Невозможно осуществлять поиск по пользователям. Однако на фоне всех предоставляемых приложением возможностей эти пробелы являются незначительными.

Данное приложение вряд ли подойдет начинающему менеджерам проектов или команде разработчиков, так как содержит большое количество функционала и может быть раскрыто не полностью. Оно ориентированно прежде всего на опытных специалистов, которые имеют большой опыт коммерческой разработки. Таким образом, «Jira» – это самое функциональное приложение из всех, что были описаны ранее, но за счет этого одновременно и более трудное в освоении для начинающих. Поэтому основной ориентир в ходе разработке будет установлен на данный аналог.

* 1. Патентный поиск

В данном подразделе представлены результаты патентного исследования. По итогу проведенного поиска были выявлены патенты, имеющие схожий с реализуемым приложением функционал. Они представлены ниже в таблицах 1.1 и 1.2.

Таблица 1.1 – Описание патента №1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Dynamic digital calendar day kanban board» | US10496962B1 | 03.12.2019 | Bryan Gregory Bartels, Patrick Humpal |

В данном патенте описывается технология управлениями задачами на основе использования традиционных методов, таких как, например, использование списков или календарных отметок, в сочетании с более гибкими методами, то есть Kanban-доски. Патент предлагает отображение задач гибкое, наглядное, простое, но одновременно и строго структурированное.

Технология позволяет группировать задачи по секциям, где каждая секция относится к конкретному календарному, рабочему или иному дню. Каждая из этих секций имеет свои подсекции, олицетворяющие прогресс или статус задач. Таким образом, каждая задача находится в такой подсекции согласно своему статусу и дню ее планируемого завершения. Изобретение позволяет дублировать в секциях частые и повторяющиеся задачи. Дополнительно технология предполагает измерений изменений, вносимых пользователями в задачи, для последующего присвоения пользователям баллов, отображающих их продуктивность. Также технология способна переприкреплять задачу на следующий день, если задача должна была быть завершена в текущий день, но ее статус еще не является завершенным [4].

Таблица 1.2 – Описание патента №2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| «Kanban-based work processing device and work processing method» | US20220327486A1 | 13.10.2022 | Jae Ho Choi |

Данный патент описывает процесс обработки задач, который позволяет пользователю увидеть ход выполнения проекта через Kanban-доску, в которой к каждой задаче создается чат для общения между коллегами насчет задачи и проведения системных оповещений и напоминаний. Данный чат будет автоматически оповещать сотрудников о создании новой задачи, об изменении статуса задачи, о том, что срок выполнения задачи подходит к концу и др. Если неавторизованный пользователь будет приглашен в чат, то ему будет выдан доступ к Kanban-доске и задачам. Итого данный метод позволяет следить за эффективностью и прогрессом работы [5].

* 1. Выводы по разделу

В данном разделе были поставлены задачи для выполнения, были рассмотрены и проанализированы аналогичные существующие на рынке программные решения, а также был проведен патентный поиск по теме разрабатываемого дипломного программного модуля и необходимого функционала.

В процессе обзора и анализа в данной главе нескольких аналогичных разрабатываемому программному средству приложений были отмечены достоинства и недостатки каждого из аналогов, выбраны положительные стороны для дальнейшей разработки и проанализированы недостатки. Были обнаружены недостатки, которые присутствуют во всех рассмотренных аналогах, поэтому при разработке приложения им особенно будет уделено внимания для их недопущения. Было выявлено, что ни один из рассмотренных аналогов не предназначен для использования в качестве полноценного корпоративного приложения с возможностью управления проектами, поэтому разработку данного проекта можно считать нужной и целесообразной.

Было также найдено и рассмотрено несколько зарегистрированных патентов, описывающих и предоставляющих общие вспомогательные возможности для реализации функционала программных средств по теме управления IT-проектами с использованием такой методологии как Kanban.

Таким образом, было выявлено, что разработка данного проекта является целесообразной, а также был определен основной спектр функциональных возможностей разрабатываемого проекта, на которые в будущем необходимо сделать упор, для выделения приложения на фоне его конкурентов.

1. Проектирование программного средства

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

2 Проектирование

веб-приложения

Лит.

Листов

18

*74218051, 2023*

У

Николайчук А.Н.

Проектирование веб-приложения является важным этапом в разработке, который определяет будущую работу системы. Правильно спроектированное веб-приложение обеспечивает не только эффективную работу системы, но и улучшает взаимодействие пользователей с информацией, что в свою очередь приводит к увеличению доходов бизнеса и повышению удовлетворенности клиентов. Таким образом, проектирование веб-приложения является важным этапом в разработке информационных систем и требует особого внимания со стороны разработчиков.

Сначала необходимо определиться с подходящими языками программирования и фреймворками для разработки серверной и клиентской частей приложения. Важно выбрать язык программирования, который подходит для решения поставленных задач, обладает необходимыми возможностями и инструментами для работы с выбранными технологиями и позволяет обеспечить высокую производительность и качество кода. Необходимо выбрать такие языки программирования, которые используются для разработки веб-приложений.

Далее будет спроектирована база данных. Правильное проектирование базы данных позволяет обеспечить гибкость, масштабируемость и легкость в разработке и поддержке системы, что является важным фактором для успеха проекта. Также важно подобрать такую базу данных, которая будет корректно и без сбоев взаимодействовать с серверной часть приложения.

Стоит также отметить, что для упрощения разработки приложения должна была добавлена возможность виртуализации или контейнеризации, чтобы можно было легко и быстро воссоздать среду окружения, в котором будет размещено приложение. Для этого было выбрано средство контейнеризации Docker.

* 1. Диаграмма вариантов использования

Диаграмма использования представляет собой графическое изображение взаимодействия между системой и ее окружением, и отражает то, как пользователи взаимодействуют с системой для выполнения конкретных задач. Диаграммы использования широко используются в разработке ПО для определения требований, моделирования функциональности и тестирования системы.

Диаграмма использования состоит из двух основных элементов: актеров и вариантов использования. Актеры – это пользователи или другие системы, которые будут взаимодействовать с системой. Варианты использования – это действия, которые пользователи могут выполнять в системе, чтобы достичь своих целей. Каждый вариант использования описывает сценарий, в котором пользователь взаимодействует с системой, и описывает, как система должна реагировать на эти действия.

Диаграмма использования позволяет определить требования к системе и спланировать ее функциональность до начала фактической разработки. Это позволяет убедиться, что разработка системы отвечает требованиям пользователей и эффективно решает их задачи. Кроме того, диаграмма использования может использоваться в качестве средства коммуникации между разработчиками и заказчиками для обеспечения взаимопонимания и улучшения качества конечного продукта.

Диаграмму вариантов использования для дипломного проекта можно увидеть в приложении А. На данной диаграмме есть 4 роли: сотрудник, менеджер, администратор и заказчик. Любой пользователь в приложении может авторизовываться, осуществлять поиск по всем сотрудникам, просматривать профили сотрудников, осуществлять поиск по проектам, просматривать детальную информацию о проекте, просматривать детальную информацию о задачах, редактировать свой профиль.

Пользователь с ролью заказчика может также оставлять заявки на выполнения проекта, прикрепляя к заполняемой формы файл с описанием проекта.

Пользователь с ролью сотрудника может осуществлять работу с задачами, а именно: создавать задачи, редактировать задачи, логировать время, потраченное на выполнение задачи. Также любой сотрудник выполняет работу с комментариями: создание комментариев, редактирование и удаление своих комментариев.

Пользователь с ролью менеджера могут выполнять все, что выполняет пользователь с ролью сотрудника, а также просматривать существующие заявки, оставленные заказчиками, и на их основе создавать новые проекты, назначать сотрудников на создаваемые проекты, редактировать проекты, блокировать заявки.

Пользователь с ролью администратора может выполнять все те же действия, что и пользователь с ролью менеджера, а также создавать аккаунты сотрудников, деактивировать аккаунты сотрудников, блокировать и разблокировать аккаунты сотрудников, изменять и удалять чужие комментарии.

* 1. Выбор средств реализации

При разработке приложения необходимо выбрать языки программирования, инструменты и технологии, подходящие для реализации клиентской и серверной частей приложения. Правильный выбор средств реализации может значительно упростить и ускорить процесс разработки и реализации проекта, а также снизить риски возникновения проблем в работе готовой системы. Неправильный выбор средств может привести к проблемам в процессе разработки и реализации проекта, а также к неполадкам в работе готовой системы. Эта глава является ключевой для успешной реализации проекта и представляет собой основу для последующих разделов работы.

* + 1. Основные языки программирования

Перед началом разработки веб-приложения необходимо выбрать языки программирования для написания клиентской и серверной частей проекта.

Java – это распространенный и широко используемый язык программирования, с большой базой пользователей и многочисленными приложениями в различных отраслях. Одно из его основных преимуществ – универсальность, поскольку он может использоваться как для разработки приложений, так и для веб-разработки. Он является платформонезависимым, что означает, что программы, написанные на Java, могут работать на любой операционной системе, их не надо перекомпилировать для разных систем. Согласно данным исследования Statista за 2022 год, в настоящее время доля Java на рынке среди всех языков программирования составляет 34%, что делает его вторым по популярности языком после JavaScript (40%) [6].

При разработке проекта будет использоваться Java версии 8. Это очень значительная версия языка с точки зрения функциональности и улучшений API. На данный момент существует уже множество новых версий Java, но есть определенные причины, по которым Java 8 до сих пор может быть предпочтительнее новых версий. Java 8 существует уже довольно долгое время, и поэтому она является очень стабильной и зрелой версией языка. Она прошла множество тестов и считается надежной и стабильной в производственных средах. Java 8 широко поддерживается сторонними библиотеками и фреймворками. Если вы используете старое программное обеспечение, которое еще не поддерживает новые версии Java, восьмая версия может быть единственным вариантом. Java 8 имеет хорошую производительность, и многие оптимизации были выполнены в последующих версиях. Однако в некоторых случаях Java 8 может работать быстрее, особенно когда используется API Stream. Также Java 8 существует уже долго, и многие разработчики знакомы с ее синтаксисом и возможностями. Если уже есть код, написанный на Java 8, может быть эффективнее продолжать использовать ее, чем переходить на новую версию.

При разработке клиентской части использовался язык программирования JavaScript. Это один из самых популярных языков программирования, который широко используется веб-разработкой. Он может быть использован для создания динамических и интерактивных пользовательских интерфейсов, а также для создания более сложных приложений. JavaScript поддерживается всеми основными браузерами. Для упрощения разработки и создания асинхронных интерфейсов вместе с JavaScript будет использоваться библиотека, описанная далее.

* + 1. Фреймворки и библиотеки

Фреймворки – это программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули, а все специфичные для проекта компоненты реализуются разработчиком на их основе. Тем самым достигается не только высокая скорость разработки, но и довольно большая производительность, хорошая надежность, простота реализации решений [7].

Библиотека – это набор функций, которые решают конкретную задачу в программе. Она может применять шаблоны сообщений, ранее скомпилированный код, классы или подпрограммы – и использоваться много раз. В языках программирования есть стандартные библиотеки, но разработчик также может создать свою.

При разработке серверной части использовался Spring – это популярный фреймворк для создания веб-приложений с использованием Java. Это часть фреймворка Spring, которая представляет собой набор инструментов и библиотек для создания приложений корпоративного уровня. Spring упрощает создание автономных приложений производственного уровня, которые можно легко развернуть и запустить с минимальной конфигурацией. Одним из самых больших преимуществ Spring Boot является его способность автоматически настраивать ваше приложение на основе зависимостей, которые вы добавили в свой проект. Эта функция экономит много времени и усилий разработчикам, так как устраняет необходимость в ручной настройке. В результате разработчики могут сосредоточиться на написании кода и реализации функций, а не тратить часы на настройку приложения [8]. При разработке приложения использовалась версия фреймворка 2.7.3.

Для разработки клиентской части был выбран React – это JavaScript-библиотека для создания пользовательских интерфейсов, которая обеспечивает эффективность, гибкость, удобство использования, кроссплатформенность и имеет большое сообщество разработчиков. Выбор React для фронтенд-разработки обеспечивает быстрое создание сложных пользовательских интерфейсов, упрощение и ускорение процесса разработки, а также улучшение производительности и эффективности работы приложения. Очень важное преимущество React – это наличие множества написанных для него библиотек, с помощью которых можно просто и быстро разработать нужное. При разработке приложения использовался React 18.2.0. На момент написания проекта это самая свежая и самая стабильная версия. Использование новейших версий позволяет использовать новые возможности, которые могут значительно упростить и ускорить процесс разработки. Новые версии языков программирования могут предоставлять улучшенные механизмы оптимизации, что может привести к значительному увеличению производительности приложений. Новые версии языков программирования обычно имеют большую поддержку со стороны сообщества разработчиков. Это может включать большое количество библиотек и фреймворков, которые могут значительно упростить разработку. Именно поэтому при разработке приложения была выбрана новейшая версия React.

* + 1. Система управления базами данных

База данных является важнейшей частью любого приложения. Все данные, с которыми взаимодействуют пользователи и само приложение хранятся именно в ней. Если с базой данных что-то случится, то приложение будет непригодно к использованию. Поэтому важно спроектировать надежную базу данных.

На данный момент существуют различные типы баз данных: реляционные, сетевые, иерархические и др. Для разработки проекта было решено использовать реляционную базу данных. Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором (первичным ключом), а строки из нескольких таблиц могут быть связаны внешними ключами [9].

Среди множества существующих реляционных баз данных была выбрана MySQL – это популярная реляционная база данных, которая широко используется в различных типах приложений, от простых веб-сайтов до крупных корпоративных систем. Использование MySQL имеет следующие преимущества:

* бесплатная и открытая: MySQL доступна для бесплатного использования и имеет открытый исходный код, что делает ее доступной и гибкой для проектов;
* высокая производительность: MySQL обеспечивает высокую скорость выполнения запросов и обработки данных, что делает его отличным выбором для приложений с большим объемом данных;
* надежность и безопасность: MySQL является надежной базой данных, которая обеспечивает защиту от утечек данных и взлома. Он имеет ряд функций безопасности, таких как шифрование данных, контроль доступа и аудит.
* гибкость: MySQL легко интегрируется с различными языками программирования, фреймворками и операционными системами.

Таким образом, MySQL – это надежная, безопасная и производительная база данных, которая хорошо подходит для широкого спектра приложений. При разработке была использована версия 8.0. На момент разработки это самая крайняя версия данной базы данных. Ее основные плюсы: улучшенная производительность и оптимизация запросов, повышенная безопасность и усиленная защита данных, введение оконных функций и общих выражений таблиц для более мощного анализа данных.

* + 1. Вспомогательные инструменты

С помощью JavaScript и React можно разрабатывать функции и компоненты, которые буду реагировать на действия пользователя или выполнять другую логику, а для создания непосредственно самого представления интерфейса необходимо использовать вспомогательные инструменты, а именно язык верстки HTML и язык описания внешнего вида CSS. Именно эти инструменты создают, определяют и влияют на внешний вид приложения.

Для создания красивых и современных элементов интерфейса была задействована библиотека React-Bootstrap 2.5.0, в частности для создания модальных окон и форм. Дополнительно была использована библиотека Bootstrap Icons 1.9.1 для возможности использования стильных иконок.

Для выполнения асинхронных запросов с клиентской части на серверную использовалась Axios 0.27.2 – это библиотека JavaScript для выполнения HTTP-запросов в браузере и на сервере с использованием промисов. Она позволяет легко отправлять запросы и обрабатывать ответы в удобном формате. Axios поддерживает практически все методы HTTP-запросов. Она также обеспечивает возможность установки заголовков запроса, передачи параметров запроса, обработки ошибок и другое. Axios широко используется в различных проектах на JavaScript, в том числе в связке с библиотекой React для обмена данными между клиентской и серверной частями.

Для построения графиков и диаграмм в клиентской части использовалась Chart.js 4.2.1 – это JavaScript-библиотека для создания интерактивных диаграмм и графиков на веб-страницах. Она позволяет создавать различные типы графиков, такие как линейные, столбчатые, круговые и т.д. Библиотека Chart.js предоставляет возможности для настройки отображения данных, включая цвета, шрифты, легенды и другие параметры. Она имеет открытый исходный код, хорошую документацию и поддержку сообщества, что делает ее популярным выбором для веб-разработки.

* 1. Авторизация пользователей

Безопасность приложения – это ключ к надежному, безопасному приложению и сохранности данных в нем. Необходимо обязательно не допустить в приложении доступ к конфиденциальным и уязвимым данным неавторизованных лиц.

Процесс доступа к приложениям осуществляется в 3 этапа:

* идентификация – это процедура, в результате выполнения которой для субъекта выявляется его уникальный признак, однозначно определяющий его в системе;
* аутентификация – это процедура проверки подлинности пользователя, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных;
* авторизация – это предоставление прав на выполнение действий [10].

Самый простой подход для аутентификации в приложениях – это отправка логина и пароля пользователя при каждом запросе. Однако очевидно, что такой способ небезопасен, особенно если клиент использует незащищенный протокол. Более удобным, безопасным и распространенным способом является использование токенов. Было принято решении в приложении реализовать использование JSON Web Token. JWT является популярным форматом токена авторизации в веб-приложениях по нескольким причинам. JWT содержит всю необходимую информацию для проверки подлинности пользователя, включая информацию о правах доступа. Это означает, что приложение не требует сохранения состояния на сервере, что уменьшает нагрузку на сервер. JWT может быть расширен для поддержки дополнительной информации, такой как права доступа, срок действия, идентификатор пользователя и другое. JWT позволяет подписывать токен с использованием секретного ключа, что обеспечивает дополнительный уровень безопасности. JWT поддерживается большинством языков программирования, что делает его удобным для использования в любом веб-приложении, независимо от технологий, используемых на серверной стороне. JWT легко передается между клиентом и сервером в HTTP-заголовке, параметрах URL или в теле запроса. На рисунке 2.1 представлено, каким образом осуществляется авторизация пользователей в приложении с использованием токенов.

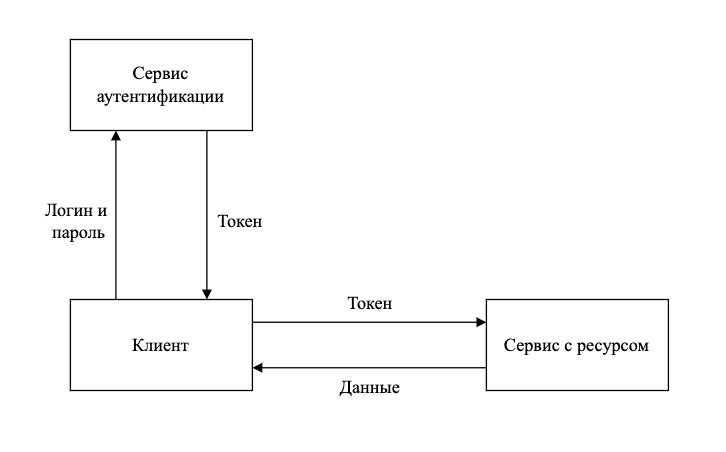


Рисунок 2.1 – Схема авторизации пользователей

Здесь описан следующий порядок действий: при первичной аутентификации пользователь отправляет свои логин и пароль и получает в ответе токен. Далее он использует этот токен для авторизации в приложении путем добавления токена в заголовки при каждой отправке запроса. Когда срок действия токена истечет, пользователь проходит процедуру аутентификации, отправляя на сервер свои логин и пароль и получая в ответ новый работоспособный токен.

Особенностью токенов является ограниченный срок жизни, то есть, когда сервер выдает новый токен, он фиксирует время его создания и записывает в служебную информацию. Если время жизни токена подошло к концу, на клиентской части пользователю будет выведено предупреждающее об этом модальное окно и пользователю буду необходимо заново авторизироваться в приложении.

* 1. Контейнеризация

Для запуска и развертывания приложения на локальной машине необходима установка различных программ и компонентов. Очевидно, что это неудобно и сложно для конечных пользователей. Есть возможность разместить приложение на облачных платформах, например, AWS или Azure, но это может быть финансово затратным и могут возникнуть проблемы с конфигурацией. Поэтому оптимальным решением является использование контейнеризации.

Технология контейнеризации – это форма виртуализации ОС, предлагающая изоляцию приложений в пользовательских пространствах (контейнерах). Все контейнеры используют одну и ту же операционную систему. Благодаря технологии контейнеризации можно запускать приложение с нужными библиотеками в типовом контейнере, который соединяется с хостом или другой внешней компонентой при помощи простого интерфейса. Все компоненты, необходимые для работы приложения (код, среда запуска, системные инструменты, библиотеки и настройки), упаковываются в один образ и могут быть использованы повторно в рамках текущей задачи или для любых других. Контейнер независим от ресурсов и архитектуры хоста. Он создает изолированную среду для приложения, не используя CPU, RAM или хранилище хостовой ОС. Все процессы идут внутри [11].

Самым популярным средством для контейнеризации является Docker. Данное средство занимает лидирующую позицию на рынке и интегрировано в облачные сервисы. Оно кроссплатформенно, это означает, что приложение, имеющее поддержку Docker, сможет быть запущено на любой ОС с предустановленным Docker-ом. Помимо обычных средств контейнеризации Docker включает в себя поддержку оркестрации контейнеров, что означает, что можно получить контроль сразу над несколькими контейнерами, управлять их запуском и так далее.

Чтобы запустить разрабатываемое приложение внутри Docker, необходимо сконфигурировать определенным файлы, а именно файлы типа Dockerfile для серверной и клиентской частей, чтобы сгенерировать их образы, а также файл docker-compose.yaml для запуска всех образов как единое целое.

Конфигурационный файл Docker, который используется для генерации контейнера клиентской части, приведен в листинге 2.1.

На основе данного файла Docker создаст образ клиентской части приложения, который является основой для создания контейнера.

|  |
| --- |
| FROM node:17-alpine3.15  COPY package.json package-lock.json ./  RUN npm install --legacy-peer-deps  COPY . .  # RUN npm run build  CMD ["npm", "start"] |

Листинг 2.1 – Dockerfile клиентской части

Поскольку все приложение состоит из нескольких частей: серверная часть, клиентская часть, база данных – необходимо сконфигурировать все docker-контейнеры для них так, чтобы они корректно взаимодействовали между собой. Этот процесс существенно облегчает использование Docker Compose – это средство для определения и запуска приложений Docker с несколькими контейнерами. Файл YAML определяет все службы, которые необходимо развернуть. Эти службы зависят либо от Dockerfile, либо от существующего образа контейнера [12]. Таким образом, для работы с Docker Compose необходимо создать дополнительный конфигурационный файл docker-compose.yaml. Полный пример данного файла показан в листинге приложении Б, а на листинге 2.2 приведен краткий пример запуска базы данных. Поскольку запуск серверной части зависит от того, запустилась ли уже база данных, то осуществляется проверка «состояния здоровья» контейнера через запросы.

|  |
| --- |
| mysql-container:  image: mysql:8.0.28  volumes:  - "./db.sql:/docker-entrypoint-initdb.d/1.sql"  - "./DML.sql:/docker-entrypoint-initdb.d/2.sql"  - dbdata:/var/lib/mysql  environment:  MYSQL\_ROOT\_PASSWORD: ${MYSQL\_ PASSWORD}  healthcheck:  test: ["CMD", 'mysqladmin', 'ping', '-h', 'localhost', '-u', 'root', '-p$MYSQL\_ROOT\_PASSWORD']  interval: 20s  timeout: 10s  retries: 10  start\_period: 30s |

Листинг 2.2 – Конфигурация базы данных в Docker Compose

Корневым элементом для описания контейнеров является тег services. Далее начинается описание каждого из контейнеров с его названия. В каждом из контейнеров указывается путь к Dockerfile, на основе которого будет создан образ, или название уже существующего образа. Также можно указать порты приложения, на одном из них приложение запускается внутри Docker, а с использованием другого можно получить доступ к приложению извне. Дополнительно можно указать проверку «здоровья» контейнера, то есть запустился ли он и успешно ли, а также указать контейнеры, от которых зависит текущий, то есть пока зависящие контейнеры не запустятся, текущий не будет запущен. Для создания контейнера с базой данных использовался уже существующий образ, который Docker загружает с DockerHub – это официальное общедоступное хранилище всех образов, куда также могут быть загружены и пользовательские образы. Чтобы при выключении или перезапусках Docker база данных не теряла хранящиеся в ней данные, необходимо указать volumes – это инструмент, который отключает привязку данных к жизненному циклу контейнера, позволяя получить доступ к контейнерным данным в любой момент.

Таким образом, с использованием контейнеризации можно легко запускать любые приложения на любой инфраструктуре и операционной системе.

* 1. Проектирование базы данных

Как было сказано ранее, для разрабатываемого проекта была выбрана реляционная база данных MySQL 8.0. База данных запускается в качестве Docker-контейнера, конфигурация приведена в листинге 2.2. Структуру базу данных, таблицы, поля, связи можно увидеть в приложении В и на рисунке 2.2.

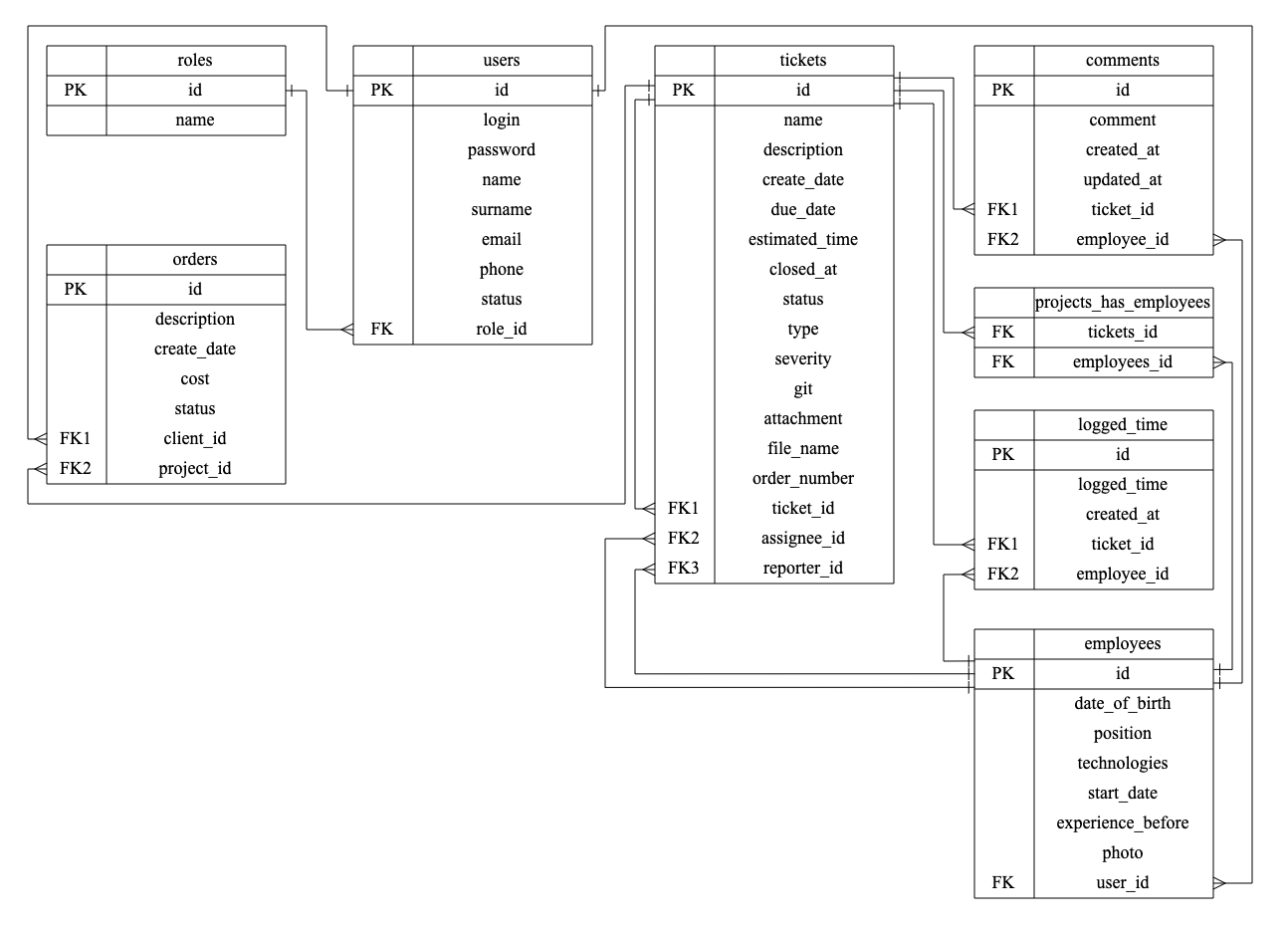


Рисунок 2.2 – Схема базы данных приложения

В базе данных существует 8 таблиц: 7 основных для хранения информации о проекте, 1 – для реализации отношения между сущностями «многие-ко-многим».

Далее будет подробно описана структура и содержимое каждой таблицы.

* + 1. Таблица roles

Данная таблица содержит существующие в приложении роли.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец name. Хранит название роли.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле name имеет ограничение, которое не допускает незаполнение данного поля. Также все значения поля name должны быть уникальными.

* + 1. Таблица users

Данная таблица содержит информацию о существующих пользователях.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец role\_id. Хранит идентификатор роли, которой наделен пользователь;
* столбец login. Хранит логин профиля пользователя;
* столбец password. Хранит пароль профиля пользователя;
* столбец name. Хранит имя пользователя;
* столбец surname. Хранит фамилию пользователя;
* столбец email. Хранит адрес электронной почты пользователя;
* столбец phone. Хранит номер телефона пользователя;
* столбец status. Хранит статус профиля пользователя.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле role\_id является внешним ключом для связи с таблицей roles. Поля role\_id, login, password, name, surname, status имеют ограничение, которое не допускает незаполнение данных полей. Также все значения поля login должны быть уникальными.

* + 1. Таблица employees

Данная таблица содержит дополнительную информацию о пользователях, которые являются сотрудниками компании.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец user\_id. Хранит идентификатор сотрудника;
* столбец date\_of\_birth. Хранит дату рождения сотрудника;
* столбец position. Хранит должность сотрудника;
* столбец technologies. Хранит список навыков, которыми владеет сотрудник;
* столбец start\_date. Хранит дату принятия сотрудника на работу в компанию;
* столбец experience\_before. Хранит количество лет опыта, которые были у сотрудника на момент трудоустройства в компанию;
* столбец photo. Хранит фотографию сотрудника.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле user\_id является внешним ключом для связи с таблицей users. Поля user\_id, position, start\_date имеют ограничение, которое не допускает незаполнение данных полей.

* + 1. Таблица tickets

Данная таблица хранит информацию о всех существующих проектах и задачах.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец assignee\_id. Хранит идентификатор сотрудника, который является ответственным за выполнение данной задачи или проекта;
* столбец reporter\_id. Хранит идентификатор сотрудника, который создал данную задачу или создал данный проект;
* столбец ticket\_id. Хранит идентификатор проекта, к которому относится задача. В случае проекта это поле пустое;
* столбец name. Хранит название задачи или проекта;
* столбец description. Хранит описание задачи или проекта;
* столбец create\_date. Хранит дату создания задачи или проекта;
* столбец due\_date. Хранит дату, когда задача или проект должны завершится;
* столбец estimated\_time. Хранит примерное количество часов, которое необходимо на выполнение данной задачи или проекта;
* столбец closed\_at. Дата и время, когда задача была завершена;
* столбец status. Хранит статус задачи или проекта;
* столбец type. Хранит тип задачи или проекта;
* столбец severity. Хранит уровень серьезности задачи или проекта;
* столбец git. Хранит ссылку на VCS задачи или проекта;
* столбец attachment. Хранит прикрепленный к задаче или проекту файл;
* столбец file\_name. Хранит название прикрепленного файла;
* столбец order\_number. Хранит порядковый номер для вывода на дэшборде.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поля assignee\_id и reporter\_id являются внешними ключами для связи с таблицей employees. Поле ticket\_id является внешним ключом для связи таблица с самой собой. Поля assignee\_id, reporter\_id, name, create\_date, status, type, имеют ограничение, которое не допускает незаполнение данных полей.

* + 1. Таблица projects\_has\_employees

Связующая таблица между сущностями Ticket и Employee для корректной реализации отношения «многие ко многим» и хранит информацию о том, какой сотрудник работает или работал на каких проектах.

В ее состав входят следующие столбцы:

* столбец ticket\_id. Хранит уникальный идентификатор проекта;
* столбец employee\_id. Хранит уникальный идентификатор сотрудника.

Данная таблица содержит ограничение по внешнему ключу к таблице employees по столбцу employee\_id, а также к таблице tickets по столбцу ticket\_id соответственно. Уникальный идентификатор данной записи является составным и получается автоматически на основании идентификатора проекта и сотрудника.

* + 1. Таблица comments

Данная таблица содержит информацию о комментариях к задачам.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор комментария;
* столбец comment. Хранит содержание комментария к задаче;
* столбец ticket\_id. Хранит идентификатор задачи, под которой пользователем однажды был оставлен комментарий;
* столбец employee\_id. Хранит идентификатор сотрудника, которым был написан к задаче данный комментарий;
* столбец created\_at. Хранит дату, когда был создан комментарий;
* столбец updated\_at. Хранит дату, когда пользователем крайний раз был изменен данный комментарий к задаче.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле ticket\_id является внешним ключом для связи с таблицей tickets. Поле employee\_id является внешним ключом для связи с таблицей employees. Поле comment имеет ограничение, которое не допускает незаполнение данного поля.

* + 1. Таблица logged\_time

Данная таблица содержит информацию о залогированном времени у задачи.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец logged\_time. Хранит количество часов в виде числа с плавающей точкой, потраченных на выполнение данной задачи;
* столбец created\_at. Хранит дату и время, когда было залогировано время;
* столбец ticket\_id. Хранит идентификатор задачи, для которой однажды пользователем было залогировано данное время;
* столбец employee\_id. Хранит идентификатор сотрудника, который однажды залогировал время к данной задаче.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле ticket\_id является внешним ключом для связи с таблицей tickets. Поле employee\_id является внешним ключом для связи с таблицей employees. Поля logged\_time и created\_at имеет ограничение, которое не допускает незаполнение данного поля.

* + 1. Таблица orders

Данная таблица содержит информацию о всех заказах на выполнение проектов, когда-либо поступивших в компанию.

В ее состав входят следующие столбцы:

* столбец id. Хранит уникальный идентификатор записи;
* столбец client\_id. Хранит идентификатор пользователя, оставившего заказ;
* столбец project\_id. Хранит идентификатор проекта, который был впоследствии создан на основе данного заказа;
* столбец description. Хранит описание заказа;
* столбец create\_date. Хранит дату создания заказа;
* столбец cost. Хранит примерную стоимость заказа;
* столбец status. Хранит статус, в котором сейчас находит заказ.

Данная таблица содержит ограничение первичного ключа для столбца id, который также обладает автоматической генерацией значения в таблице. Поле client\_id является внешним ключом для связи с таблицей users. Поле project\_id является внешним ключом для связи с таблицей tickets. Поля client\_id, description, status имеют ограничение, которое не допускает не заполнение данных полей.

* 1. Проектирование основных алгоритмов

В данном разделе будут описаны основные алгоритмы работы приложения, которые соответствуют целям дипломного проекта.

Процесс последовательного построения алгоритма выглядел следующим образом: алгоритм сначала формулировался в «крупные» команды. Затем на каждом последующем этапе отдельные детали алгоритма уточнялись. Процесс продолжался до тех пор, пока алгоритм не состоял из команд, которые были бы понятны базовому исполнителю. Данный метод называется методом последовательного уточнения алгоритма. Этот подход к проектированию алгоритмов позволяет повысить качество и надежность разрабатываемых функциональных модулей в проекте.

Для реализации каждого алгоритма были созданы соответствующие блок-схемы. Особенностью блок-схем является то, что они представляют собой графическую интерпретацию, позволяя сделать работу алгоритма более наглядной и упростить ее понимание. Каждый блок представляет собой некоторое действие или событие, при совершении которого начинается переход к другому действию.

Всего в ходе выполнения дипломного проекта было создано четыре алгоритма, каждый из которых будет описан ниже в соответствующем подразделе.

* + 1. Алгоритм перемещения задачи на «дэшборде»

При перемещении пользователем задачи на «дэшборде» внутри программы происходит определенные процессы, а также содержатся некоторые ограничения, отвечающие за безопасность данной операции.

Данный алгоритм продемонстрирован далее на рисунке 2.3.

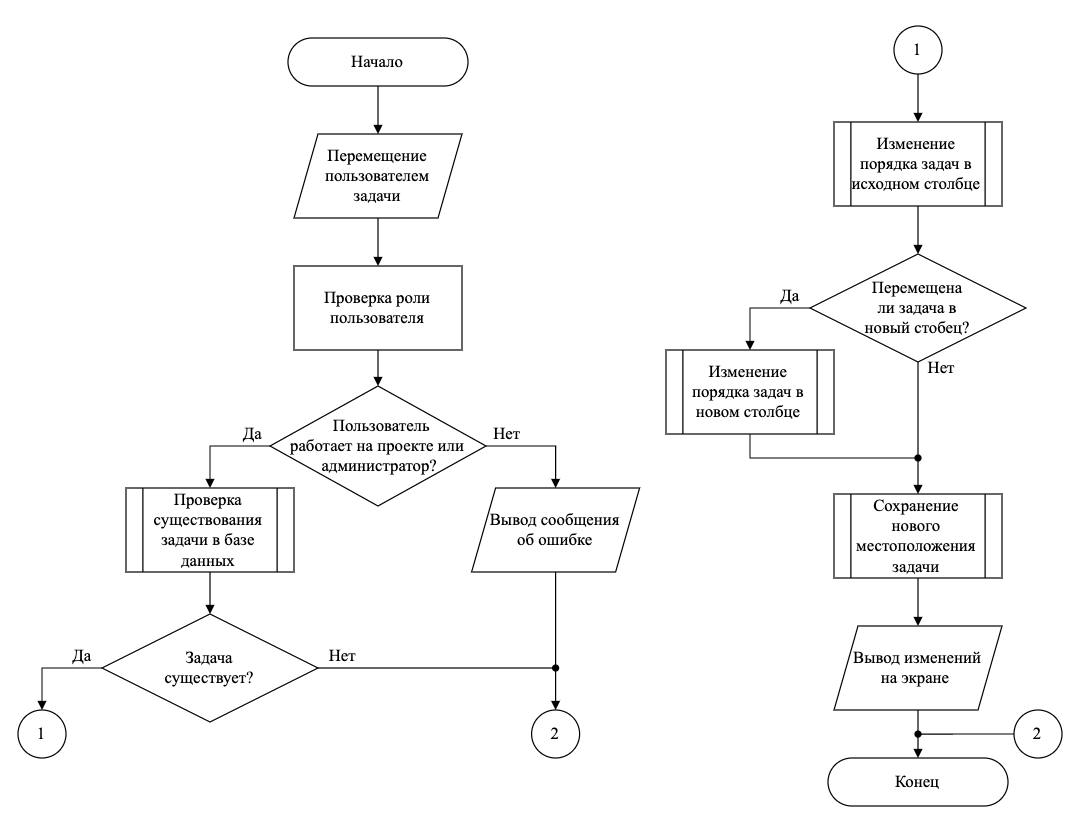


Рисунок 2.3 – Алгоритм перетаскивания задачи на «дэшборде»

При перемещении задачи на «дэшборде» происходит следующий порядок действий. Сначала пользователь перетаскивает задачу в пользовательском интерфейсе. Далее внутри приложения осуществляется проверка роли у пользователя. Если пользователь не является администратором или сотрудником проекта, то будет выведена ошибка и выполнение процесса закончится. Если права есть, то будет выполнена проверка на существование такой задачи в базе данных. Если задача не существует, то выполнение алгоритма завершится. Если задача существует, то далее будет проведено изменение порядка задач в столбце, в котором перемещаемая задача изначально находилась. Если задача была перемещена внутри этого же столбца, то новое месторасположение задачи будет сохранено, изменения будут выведена на экран, и на этом алгоритм завершится. Если же задача была перемещена в другой столбец, то далее будут выполнены все те же действия, за исключением того, что до них будет изменен порядок задач в столбце, куда задача была перемещена.

* + 1. Алгоритм изменения прикрепленного к задаче файла

При изменении прикрепленного к задаче файла внутри приложения происходит ряд процессов. Это позволяет предотвратить несанкционированное изменение данных посторонним лицом или случайные изменения.

Данный алгоритм продемонстрирован ниже на рисунке 2.4.

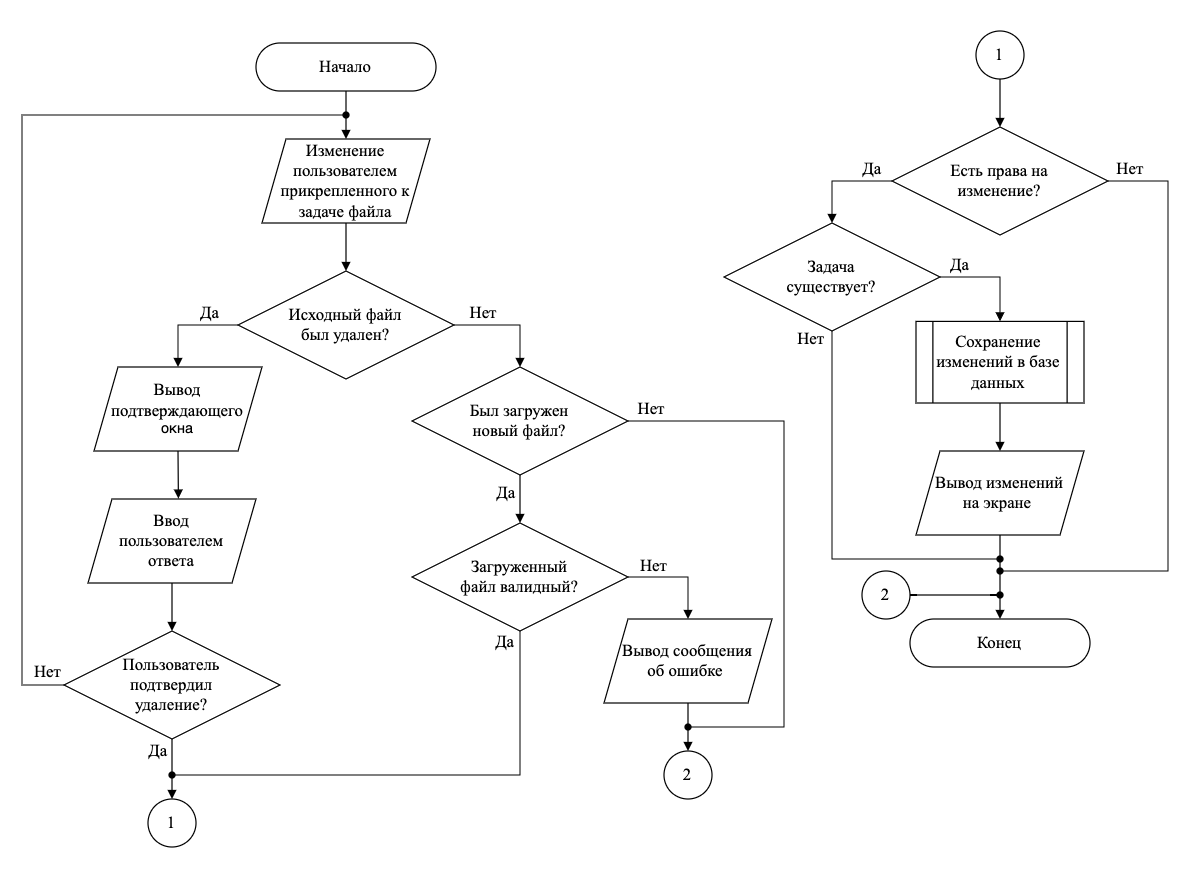


Рисунок 2.4 – Алгоритм изменения прикрепленного к задаче файла

При изменении прикрепленного к задаче файла происходит следующий ход событий. Пользователь совершает некоторые изменения с уже прикрепленным к задаче файлом. Если пользователь решил удалить данный существующий файл, то будет выведено всплывающее окно для подтверждения данного действия. Если пользователь не подтверждает удаление, то действие отменится и удаление файла не произойдет. Если же пользователь подтвердит удаление файла, то будет осуществлена проверка на наличие у пользователя необходимых прав для выполнения данного действия. Если прав у пользователя нет, то операция будет завершена. Если права у пользователя присутствуют, то будет проверено существование такой задачи в базе данных. Если задачи в базе данных не существует, текущий алгоритм будет завершен. Если задача существует, изменения будут сохранены и в итоге отображены на экране. Если пользователь не удалял файл, а загрузил новый, то этот файл будет провалидирован. Если файл невалидный, то будет выведено сообщение об ошибке и процесс завершится. Если файл валидный, то будут выполнены действия, как после шага подтверждения пользователем удаления файла. Если пользователь не удалял исходный файл и не загружал новый, то алгоритм будет завершен.

Таким образом, при изменении прикрепленного к задаче файла в приложении на фоне происходит множество действий и проверок. Все они обеспечивают безопасность и сохранность данных в приложении.

* + 1. Алгоритм авторизации заказчика в приложении

Чтобы у заказчика мог авторизоваться в приложении, необходимо выполнить целый алгоритм. Данный алгоритм продемонстрирован на рисунке 2.5.

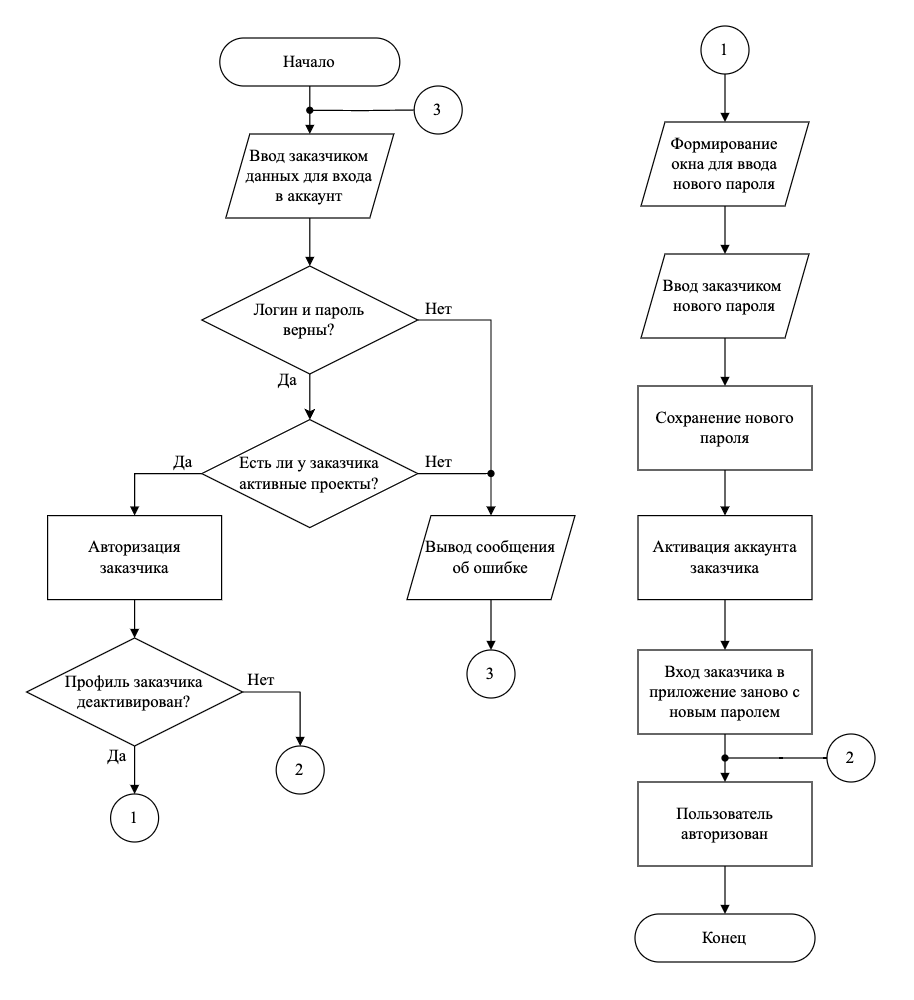


Рисунок 2.5 – Алгоритм авторизации заказчика в приложении

Чтобы заказчик мог авторизоваться в приложении, необходимо выполнить следующие шаги. Сначала заказчик переходит на страницу авторизации и вводит свои данные. Если логин или пароль неверны, будет отображена ошибка. Если данные корректны, далее будет проверено, существуют ли у заказчика проекты. Если их нету, будет выведено сообщение об ошибке и пользователь не сможет авторизоваться. Если минимум один проект на имя заказчика создан, то будет произведена авторизация пользователя. Если это первый вход заказчика в приложение, то есть если его профиль деактивирован, то на экране отобразится окно для изменения пароля. После того как пользователь введет новый пароль и сохранит его, аккаунт пользователя будет активирован и ему будет необходимо снова зайти в приложение.

* + 1. Алгоритм изменения пароля от аккаунта

Для того, чтобы изменить пароль от аккаунта, необходимо выполнять ряд шагов. Алгоритм для изменения пароля продемонстрирован на рисунке 2.6.

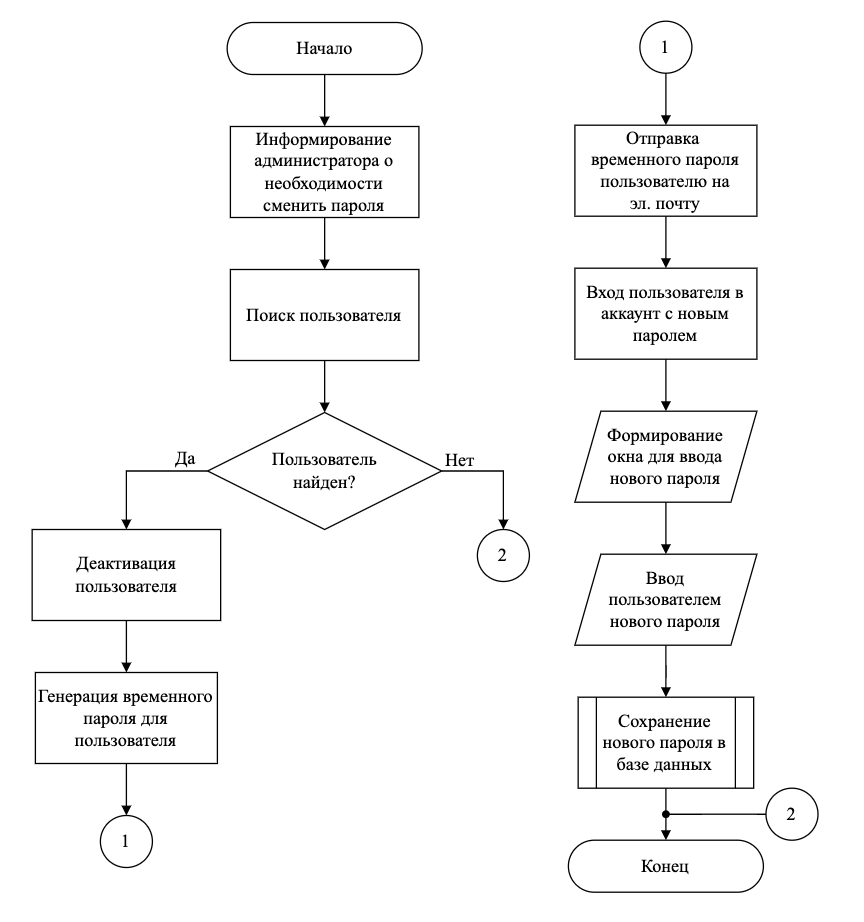


Рисунок 2.6 – Алгоритм изменения пароля

Чтобы изменить пароль от своего аккаунта, пользователю необходимо проинформировать об этом администратора. Далее администратор осуществляет в приложении поиск нужного пользователя. Если данный пользователь не найден, то операция будет завершена. Если же пользователь найден, то его аккаунт деактивируется. Это означает следующее: для пользователя будет сгенерирован новый временный пароль, который будет отправлен пользователю письмом на электронную почту. Когда пользователь получит пароль, он может, используя полученный временный пароль, авторизоваться в приложении. При первой авторизации в приложение после деактивации аккаунта на экране появится модальное окно для смены пароля, куда пользователь введет новый пароль, и он будет сохранен в базе данных. После этого пользователю будет необходимо заново авторизоваться в приложении уже с новым паролем. Таким образом, после выполнения всех описанных шагов пароль от аккаунта пользователя в приложении будет изменен.

* 1. Выводы по разделу

В разделе проектирования была построена и описана диаграмма вариантов использования. Эта диаграмма вариантов является самым общим представлением функциональных требований к системе. Варианты использования являются описаниями типичных взаимодействий между пользователями и самой системой.

В данном разделе были выбраны основные языки программирования и средства для реализации дипломного проекта

Были обоснованы: выбор средств реализации проекта, в рамках которого был выбран язык программирования, фреймворк, система управления базами данных, а также вспомогательные инструменты. В целях оптимизации, расширения функциональных возможностей и улучшению качества кода и продукта в целом, были использованы следующие фреймворки, технологии и библиотеки:

* Java 8.0 как основной язык программирования;
* Spring Framework 2.7.3 как базовый серверный фреймворк;
* React 18.2.0 как базовая библиотека для клиентской части;
* MySQL 8.0 как система управления базами данных;
* React-Bootstrap 2.5.0 для упрощения разработки визуального оформления;
* библиотека Axios 0.27.2 для упрощения отправки HTTP-запросов при помощи прикладных средств языка JavaScript;
* Docker 23.0.5 как средство контейнеризации для быстрого развертывания инфраструктуры, в которой будет выполняться проект.

Была выбрана реляционная СУБД, построена модель самой базы данных, используемой в проекте, созданы необходимые сущности для хранения данных, а также настроены все необходимые ограничения целостности и связи между сущностями для корректировки и валидации данных, используемых при работе приложения. Всего в базе данных было создано 8 таблиц, каждая из которых была подробно рассмотрена и описана в данной главе.

Было спроектировано четыре основных алгоритма для работы приложения, а именно: алгоритм перетаскивания задачи на «дэшборде», алгоритм изменения прикрепленного к задаче файла, алгоритм авторизации заказчика в приложении, алгоритм изменения пароля. Каждый из алгоритмов был подробно описан и для каждого из них была построена блок-схема.

1. Реализация программного средства

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 03.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

3 Реализация программного средства

Лит.

Листов

12

*74218051, 2023*

У

Николайчук А.Н.

Главной задачей данного дипломного проекта является создание веб-приложения для управления IT-проектами. Исходя из изучения существующих аналогов, были поставлены задачи по созданию программных компонентов для работы менеджеров и инженеров в качестве пользователей, а также по созданию интуитивно понятного, просто и комфортного интерфейса. При разработке проекта необходимо придерживаться всех особенностей в плане выбора средств реализации и технологий.

* 1. Общая архитектура приложения

На рисунке 3.1, а также в приложении И представлена схема архитектуры всего приложения в целом. Как можно увидеть, разрабатываемое веб-приложение состоит из трех частей: база данных, серверная часть и клиентская часть.

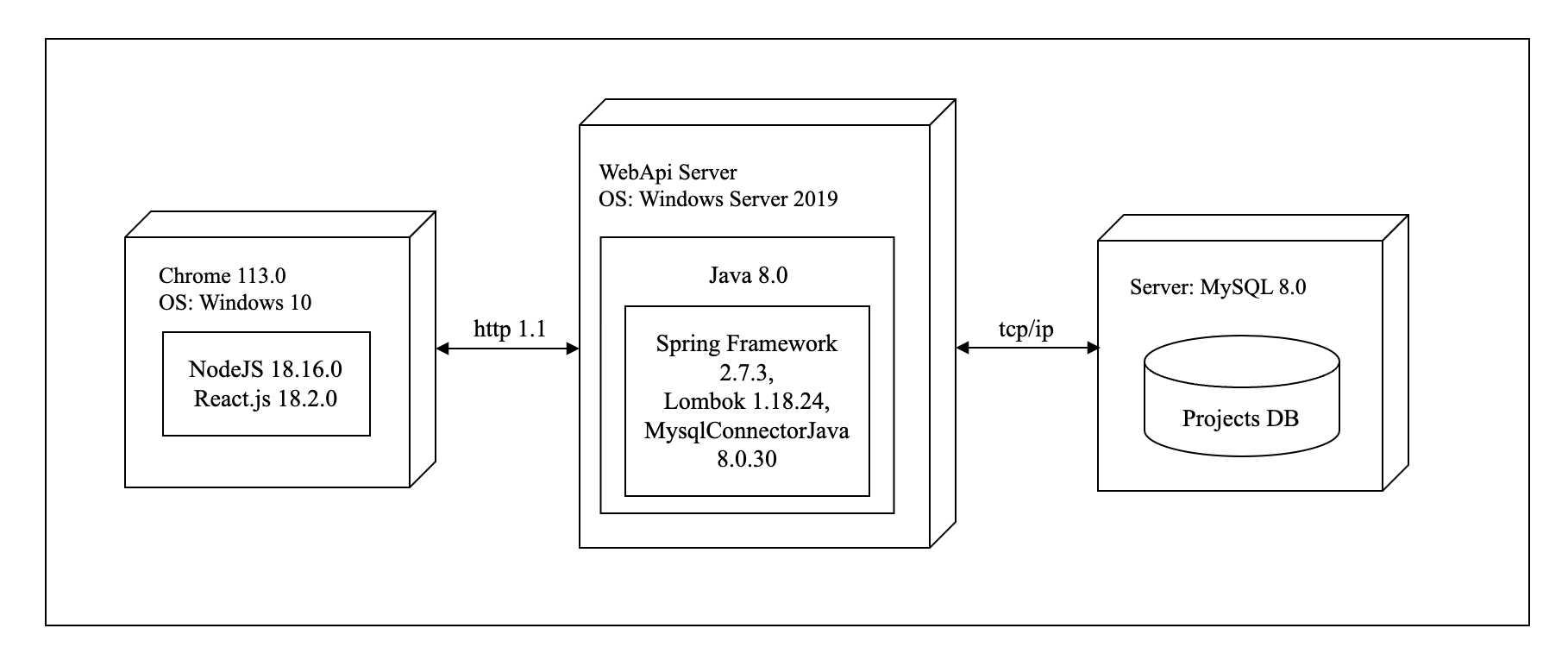


Рисунок 3.1 – Архитектура приложения

База данных приложения Projects хранится на сервере MySQL. Серверная часть приложения написана на языке программирования Java с использованием фреймворка Spring Framework и таких библиотек, как Lombok, MySqlConnectorJava и другие. Для запуска серверной части используется сервер Windows Server 2016. Клиентская часть запускается на операционной системе Windows 10 в браузере Google Chrome и написана с использованием библиотеки React.js, которая запускается на основе языка программирования NodeJS.

* 1. Разработка серверной части
     1. Установка необходимых компонентов

Перед началом создания проекта необходимо настроить среду разработки, установить все требуемые инструменты и программы. Во-первых, надо установить Java JDK. Для этого необходимо скачать пакет в соответствии с операционной системой машины с официального сайта и установить согласно инструкции [13]. После установки можно проверить корректность установки Java и доступность CLI-клиента, выполнив в терминале команду, как показано на рисунке 3.2. В качестве ответа должна быть получена установленная версия языка.

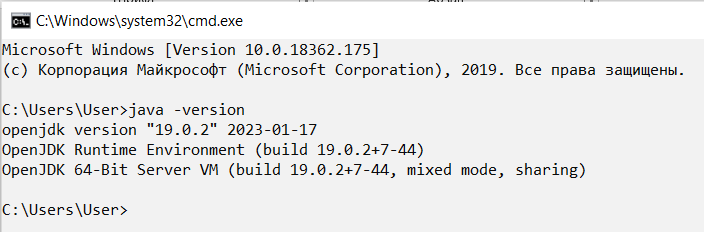


Рисунок 3.2 – Проверка Java CLI-клиента

Для разработки проекта была использована среда разработки IntelliJ IDEA. Для установки программы следует скачать мастер установки с официального сайта для нужной операционной системы и выполнить установку согласно инструкции [14]. Intellij IDEA является самой популярной средой разработки среди Java-разработчиков, поскольку предоставляет очень мощный и широкий функционал.

Дополнительно в процессе разработки использовался Maven – это инструмент для автоматизации сборки проектов. В первую очередь он позволяет легко и быстро управлять широким спектром библиотек, плагинов, используемых в приложении. Также с его помощью можно выполнять тестирование, сборку и запуск проекта и др.

При создании проекта в IntelliJ IDEA следует выбрать Maven в разделе Generator, а в поле Archetype выбрать maven-archetype-quickstart. Ранее без использования Maven (или других инструментов автоматизации) приходилось скачивать библиотеки в виде архивов, указывать к ним путь в проекте. С его же использованием достаточно найти в центральном репозитории Maven нужную библиотеку и добавить ее в pom-файл, после чего она будет автоматически скачана и добавлена в classpath приложения [15]. Для сборки проекта достаточно выполнить в терминале команду «mvn clean package», и мы получим папку со скомпилированным проектом.

Поскольку разрабатываемый проект является веб-приложением, просто так его запустить нельзя. Для этого требуется сервер, на котором будет запущено приложение. Это функцию будет выполнять Apache Tomcat – это контейнер сервлетов с открытым исходным кодом, который также выполняет функцию веб-сервера. Дополнительно устанавливать его не требуется, так как он является встроенным сервером во фреймворке Spring Boot, который будет использоваться в ходе разработки.

* + 1. Выбор архитектуры

Первые разработчики создавали программное обеспечение без архитектуры. Сначала это казалось удобным: никаких издержек, связанных с планированием, и ускоренное прототипирование. Но мере усложнения ПО теряло гибкость и управляемость, а каждое новое изменение обходилось все дороже. Это мешало развивать проект за границы, определенные изначально. Поэтому перед началом разработки необходимо выбрать приемлемую архитектуру проекта, которая будет подходить для перечисленных целей приложения и предоставлять удобство для работы.

Сперва необходимо определиться со способом взаимодействия клиентской и серверной частей. В первую очередь нужно отметить, что, поскольку в качестве клиентской части будет использоваться React, то разрабатывать веб-приложение с таким шаблонами как, например, MVC нет никакой необходимости. Поэтому для разработки нужно использовать веб-сервис, чтобы приложение могло предоставить клиенту набор интерфейсов для работы. Среди архитектур веб-сервисов наиболее лидирующую и уверенную позицию на рынке занимает REST API.

REST – это программная архитектура, которая определяет условия работы API [16]. API-интерфейсы, соответствующие архитектурному стилю REST, называются REST API. Системы, реализующие REST API, могут эффективно масштабироваться благодаря оптимизации взаимодействия между сервером и клиентом по REST. Отсутствие сохранения состояния снимает нагрузку с сервера: серверу не нужно сохранять информацию о предыдущих запросах клиента. Отлаженное кэширование частично или полностью устраняет некоторые взаимодействия между клиентом и сервером. Перечисленные функции предполагают масштабируемость и не ограничивают пропускную способность, что может привести к снижению производительности. Веб-службы RESTful поддерживают полное разделение клиента и сервера. Они упрощают и разделяют различные серверные компоненты, чтобы каждая часть могла развиваться независимо. Изменения платформы или технологии в серверном приложении не влияют на клиентское приложение. Возможность разделения функций приложения на уровни еще больше повышает гибкость. Однако, в качестве минусов можно выделить отсутствие каких-то конкретных стандартов реализации, поэтому, придется применять общепринятые практики написания сервиса для того, чтобы сделать архитектуру приложения унифицированной.

Существует два основных подхода по разработке приложения: монолитные приложения и микросервисные. Исходя из масштаба и функциональности приложения было принято решение разработать монолитное приложение и была выбрала многослойная архитектура. Этот подход работает по принципу разделения ответственностей. Приложение разделено на слои, лежащие друг на друге, и каждый из них выполняет определенную обязанность. Архитектура делит ПО на слои:

* слой представления, который содержит пользовательский интерфейс и отвечает за обеспечение хорошего пользовательского опыта;
* слой бизнес-логики, содержит бизнес-логику и бизнес-вычисления;
* слой передачи данных, который отвечает за взаимодействие с базами данных и прочую обработку информации, которая не связана с бизнесом.

Данные и элементы управления проходят через каждый слой в дизайне и передаются от одного к другому. Эта система также повышает уровень абстракции и в некоторой степени даже стабильность ПО. Благодаря разделению слоев, изменения в одном слое не влияют на другие слои, что облегчает модификацию и тестирование приложения. Многослойная архитектура позволяет обеспечить безопасность приложения, разделяя данные и логику работы приложения на разные уровни.

После того, как была выбрана и утверждена архитектура для разработки веб-сервиса, можно приступить к ее реализации. По итогу была получена общая модель архитектуры серверной части проекта, представленная на рисунке 3.3.

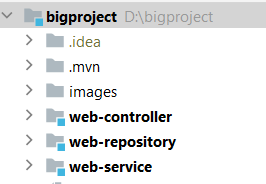


Рисунок 3.3 – Общая архитектура серверной части приложения

Как и в классической модели многослойной архитектуры, приложение содержит три слоя: слой контроллеров, слой сервисов и слой репозиториев.

Слой контроллеров содержит в себе логику обработки запросов, которые приходят от пользователя или других компонентов системы. Этот слой отвечает за прием запросов, извлечение необходимых данных и передачу их в слой бизнес-логики для обработки. Кроме этого, слой контроллеров также включает в себя валидацию данных, введенных пользователем, обработку ошибок, возникающих в процессе выполнения запросов, определение прав доступа пользователей к ресурсам системы, маршрутизацию запросов, то есть определение, какой контроллер и метод должен быть вызван для обработки каждого запроса и конвертирование доменных сущностей в DTO и наоборот. Листинг классов, отвечающих за осуществления безопасности в проекте, можно просмотреть в приложении К.

Слой сервисов является посредником между контроллерами и слоем доступа к данным. В слое сервисов выполняются операции с данными, связанные с бизнес-правилами и логикой приложения. Важно, что в этом слое нет операций, связанных с доступом к данным, так как это нарушает принципы многослойной архитектуры.

Слой репозиториев содержит логику доступа к данным. Этот слой предназначен для работы с базой данных и его основная задача – скрыть детали реализации доступа к данным от других слоев приложения. Слой репозиториев определяет интерфейсы для доступа к данным, которые реализуются в классах репозиториев. Эти классы обычно содержат методы для выполнения CRUD-операций с объектами доменной модели, а также другие методы для получения и обновления данных.

Рассмотрим архитектуру серверной части более детально. Подробная структура серверной части представлена на рисунке 3.4.

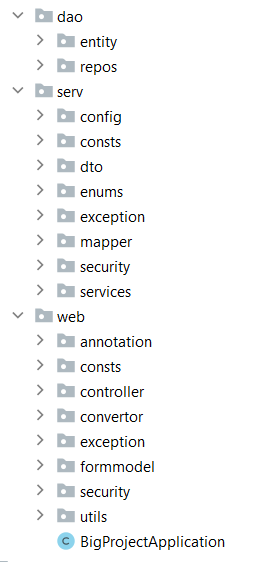


Рисунок 3.4 – Подробная архитектура серверной части приложения

В таблице 3.1 приведены все директории, существующие в проекте, а также назначение в нем каждой из них.

Таблица 3.1 – Описание структуры серверной части проекта

|  |  |
| --- | --- |
| Директория | Описание |
| 1 | 2 |
| dao | Корневая директория уровня репозиториев |
| entity | Содержит сущности для работы с базой данных |
| repos | Содержит репозитории для работы с базой данных |
| serv | Корневая директория уровня сервисов |
| config | Содержит конфигурации для уровня сервисов |
| dto | Содержит объекты передачи данных (DTO) |
| enums | Содержит перечисления |
| exception | Содержит исключения для уровня сервисов |
| mapper | Содержит реализации конвертеров из сущностей в DTO и наоборот |
| security | Содержит классы для обеспечения безопасности в приложении |
| services | Содержит имплементации сервисов |
| web | Корневая директория уровня контроллеров |
| annotation | Содержит пользовательские аннотации для валидации |
| consts | Содержит классы, хранящие в себя константы |
| controller | Содержит имплементации контроллеров |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| 1 | 2 |
| convertor | Содержит класс, для конвертации объектов, пришедших их форм |
| exception | Содержит исключения для уровня контролеров |
| formmodel | Содержит модели объектов, описывающие формы |
| security | Содержит конфигурации для безопасности приложения |
| utils | Содержит вспомогательные классы |

Таким образом, созданная структура соответствует всем необходимым требованиям приложения и предоставляет разработчикам удобство для работы с проектом, а особенно внесения изменений в нем.

* + 1. Разработка контроллеров

Контроллеры в веб-приложении являются одними из самых важных элементов для разработки, поскольку служат входными точками в серверную часть для запросов. Всего для обработки пользовательских запросов было создано восемь классовконтроллеров, как показано на рисунке 3.5.

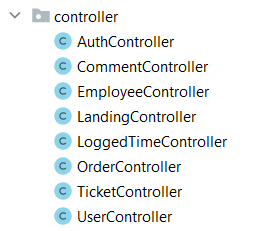


Рисунок 3.5 – Разработанные контроллеры

Пример программной реализации контроллера LoggedTimeController, отвечающего за работу с залогированным временем, показан в приложении Б.

Каждый контроллер отвечает за работу с той сущностью, которая хранится в его названии, однако стоит сделать отдельное внимание на LandingController. Все приложение реализует REST архитектуру за исключением данного контроллера, так как он единственный реализует MVC. Данный контроллер позволяет получить доступ к приветственной странице для клиентов, чтобы они могли оставлять заявки на выполнение проектов. Данная страница намеренно отделена от остального пользовательского интерфейса, поскольку разрабатываемое приложение является внутренним для конкретной компании, и было необходимо разделить его с приветственной страницей, которая является общедоступной.

Контроллеры принимают пользовательские запросы и вызывают соответствующий метод в классе-сервисе. Также на уровне контроллеров выполняется преобразование объекта передачи данных в объект сущности и наоборот.

* + 1. Разработка сервисов

Как было описано ранее, основная задача сервисов, заниматься обработкой данных: вызывать инфраструктурный уровень для чтения и обработки данных в базе данных, а также может включать в себя логическую часть по преобразованию данных в нужный формат и реализации соответствующих алгоритмов.

Всего было разработано семь сервисов, как продемонстрировано на рисунке 3.6, каждый из которых вызывается контроллером или другим сервисом.

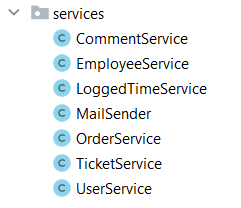


Рисунок 3.6 – Разработанные сервисы

Пример программной реализации сервиса LoggedTimeService для работы с сущностью, отвечающей за залогированное время, показан в приложении Б.

Принцип работы каждого из сервисов устроен следующим образом: данные он получает от уровня представления, или же контроллера. Затем сервис начинает обработку этих данных согласно алгоритмам. Здесь сервис имеет право вызывать инфраструктурный уровень для чтения и записи информации.

Особое внимание следует уделить сервису MailSender, поскольку если все сервисы отвечают за соответствующую их имени сущность и выполняют действия именно с ней, то данный сервис реализует интерфейс для отправки сообщений по электронной почте согласно переданным параметрам.

* + 1. Разработка репозиториев

Одной из начальных частей проекта является создание моделей и репозиториев для взаимодействия с уровнем базы данных. Чтобы иметь возможность работать с базой данных на языке Java, существует специальная ORM «Hibernate», которая позволяет взаимодействовать с данными, полученными из внешнего источника в формате Java объектов. Однако данный фреймворк будет использоваться не напрямую, а через Spring Data, поскольку Spring Data использует Hibernate в качестве одного из провайдеров сохранности данных, а также предоставляет абстракцию над Hibernate, что позволяет более удобно работать с ним. Также Spring Data поддерживает другие провайдеры, что дает возможность выбора наиболее подходящего инструмента.

При реализации проекта было решено сначала спроектировать модель базы данных, используя MySQL Workbench, и создать ее, а затем на основе уже существующей базы данных создать соответствующие сущности в проекте.

Всего были созданы семь репозиториев и семь соответствующих им сущностей, представленных на рисунке 3.7.

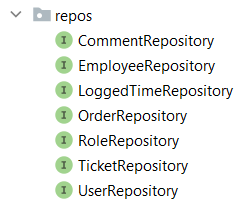


Рисунок 3.7 – Разработанные репозитории

Листинг репозитория LoggedTimeRepository представлен в приложении Б.

Для таких понятий, как сущность и репозиторий, миграция можно дать следующие определения, приведенные ниже.

Сущность – это объект в приложении, который имеет свои уникальные характеристики, идентификаторы и связи с другими объектами. Сущность обычно отображается на таблицу в базе данных, где каждое поле таблицы соответствует атрибуту сущности, а каждая запись в таблице соответствует экземпляру сущности.

В контексте разработки программного обеспечения, репозиторий – это класс или интерфейс, который отвечает за доступ и управление данными в базе данных. Он обеспечивает абстракцию между приложением и базой данных, предоставляя методы для сохранения, обновления, удаления и получения данных из базы.

* 1. Разработка клиентской части
     1. Установка необходимых компонентов

Перед началом разработки проекта, с которым непосредственно будет взаимодействовать пользователь, нужно подготовить все необходимые инструменты для работы. Прежде всего, нужно установить платформу NodeJS. Она необходима, так как react-приложение запускаются на ней.

Чтобы установить эту платформу, достаточно перейти на главную страницу официального сайта и скачать рекомендуемую версию приложения, которая является самой надежной и стабильной в работе, и выполнить все шаги согласно инструкции. Чтобы убедиться, что версия NodeJS успешно работает, достаточно выполнить в терминале команду: «node --version». Результат выполнения показан далее на рисунке 3.8. Данная команда выведет на экран версию текущей установленной на компьютере платформы NodeJS. Вместе с ней на компьютер также устанавливается пакетный менеджер npm. Именно он будет использоваться для запуска приложения и управления библиотеками в приложении. Успешность его установки проверяется аналогичной командой «npm --version» в терминале.

Теперь можно приступать как работе с самим react-приложением.

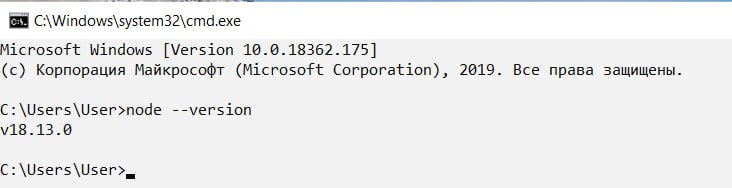


Рисунок 3.8 – Проверка NodeJS CLI-клиента

После этого можно приступить к созданию проекта. Чтобы это сделать, существует специальная утилита, разработанная компанией Facebook, позволяющая создавать приложение на React без дополнительных сложностей и конфигурации проекта. Эта утилита выглядит следующим образом: «npx create-react-app <название\_проекта>». Это означает, что будет автоматически создан проект с использованием фреймворка React. Чтобы запустить проект, достаточно перейти в директорию с проектом и написать команду «npm start». Для работы любого проекта на платформе NodeJS обязательно присутствует конфигурационный файл package.json, который хранит в себе всю информацию о проекте, а также скрипты для запуска. В нем уже находится скрипт с названием start, в котором прописана команда на запуск проекта. Далее можно приступить к самой реализации проекта.

* + 1. Выбор архитектуры проекта

Перед началом разработки клиентской проекта необходимо также выбрать приемлемую архитектуру, которая будет удобна и приемлема для реализации основных функциональных возможностей приложения, а также в дальнейшем будет легко расширяема и проста в освоении для новых разработчиков. Поскольку разрабатываемый проект имеет небольшие габариты и содержит в себе не более 10 основных страниц, была выбрана простая архитектура проекта, представленная далее на рисунке 3.9. Данный вид архитектуры является наиболее простым и понятным в использовании как для начинающих, так и опытных разработчиков.

В корне проекта содержатся конфигурационные файлы, директория node\_modules, в которой хранятся установленные по умолчанию библиотеки, а также установленные вручную дополнительные в ходе разработки библиотеки, и директория src, в которой содержатся все программные файлы. Файл index.js, располагающийся в корне папки src, является стартовой точкой приложения. В директории css находятся файлы, определяющие стиль приложения. Внутри js директории располагаются непосредственно сами компоненты интерфейса. Также внутри нее происходит логическое разделение файлов на директории согласно их бизнес-логике и функциональности, а в корне находятся независимые файлы, в частности файлы для маршрутизации в приложении, страница авторизации и файл, задающий навигационное меню. Файл package.json используется для хранения информации о проекте и его зависимостях. Этот файл содержит список всех пакетов и версий, необходимых для запуска приложения, а также другую конфигурационную информацию, такую как имя, версия, автор, лицензия, скрипты и т. д. Все зависимости приложения, включая react и его дополнительные модули, перечислены в package.json и могут быть установлены с помощью пакетного менеджера npm или yarn. Файл package-lock.json, с другой стороны, содержит информацию о точных версиях каждого пакета и его зависимостей, установленных в проекте. Этот файл создается при установке зависимостей с помощью npm install или yarn install. Он используется для гарантированной воспроизводимости установки зависимостей при переносе проекта на другую систему или для восстановления зависимостей при повторной установке.

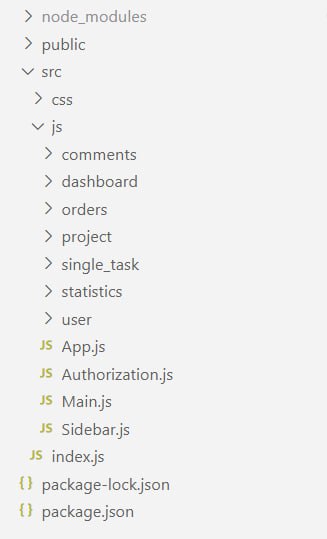


Рисунок 3.9 – Общая архитектура клиентской части

Полученная архитектура имеет ряд преимуществ. Разделение кода по функциональности позволяет легче найти и отредактировать нужный блок кода. Каждая директория имеет определенное назначение, что облегчает поиск и изменение нужных файлов. Некоторые сборщики, такие как webpack, могут использовать подобную структуру директорий по умолчанию, что позволяет быстро настроить сборку проекта. Также такая архитектура может быть легко расширена для управления другими типами файлов, например, изображениями, шрифтами и т. д.

* + 1. Маршрутизация в приложении

Для навигации по приложению для пользователей в интерфейсе представлена панель с навигационным меню. А внутри самого приложения каждому url-адресу соответствует определенный компонент. Интерфейс меню отображает пользователю только доступные в соответствие с его правами действия. Однако важно предусмотреть ситуации использования приложения недобросовестными сотрудниками или получения злоумышленниками каким-либо образом доступа к интерфейсу приложения, ведь защищенность приложения играет одну из ключевых ролей. Поэтому внутри клиентской части приложения была произведена дополнительная защита компонентов с использованием так называемых ProtectedRoutes.

Защищенные маршруты в React – это механизм, который позволяет ограничивать доступ к определенным маршрутам в приложении для пользователей, которые не авторизовались или не имеют соответствующих прав доступа. Такие маршруты защищены и требуют от пользователя предварительной аутентификации и авторизации перед тем, как он получит доступ к ним. Как правило, приложение React использует механизмы аутентификации, такие как токены или сессии, чтобы управлять доступом к защищенным маршрутам. Для создания защищенных маршрутов в React обычно используются высокоуровневые компоненты, которые обертывают защищенный компонент, проверяют статус аутентификации и авторизации пользователя и, в зависимости от этого, либо позволяют доступ к маршруту, либо перенаправляют пользователя на другую страницу, например, на страницу входа.

В данном случае был разработан защищенный маршрут со следующей логикой. Если в куках пользователя в браузере содержится авторизационный токен, причем валидный, тогда пользователь получит доступ к маршрутам приложения, иначе пользователь будет перенаправлен на страницу авторизации. Маршрут, переводящий пользователя на страницу авторизации, в свою очередь тоже обернут в дополнительный тег. Если при переходе на страницу авторизации пользователь авторизован, он будет перенаправлен на страницу приложения, на которой находился перед попыткой попасть на страницу авторизации. Таким образом, попасть на страницу авторизации авторизованный пользователь может только после полноценного выхода из приложения, используя навигационное меню. Некоторые маршруты приложения также обладают дополнительной защитой, а именно при попытках перехода на них будет осуществлена проверка на наличие у пользователя прав менеджера или администратора для посещения данных страниц. В случае отрицательного результата пользователь будет перенаправлен на страницу своего профиля. Пример маршрутизации с использованием защищенных маршрутов представлен в приложении Б.

* + 1. Валидация данных

Валидация – это процесс проверки данных на соответствие определенным правилам и требованиям, установленным для конкретного контекста. В клиентской части важно рассмотреть валидацию данных, поскольку в приложении множество форм для заполнения. Валидация происходит отдельно как на клиентской, так и на серверной части, но здесь внимание уделяется именно валидации на клиентской стороне, поскольку именно она выведет пользователю в интерфейсе сообщения о вводе некорректных данных и прервет их отправку на сервер.

Валидация происходит на основе JavaScript. Язык разметки HTML предоставляет базовые возможности для валидации, но они слишком просты и ненадежны. Поэтому в приложении используются именно возможности языка JavaScript для осуществления валидации данных при отправке формы с данными на серверную часть проекта. В приложении осуществляется проверка определенных полей на их заполненность и на длину текста в них. В случае несоответствия данных заданным условиям, пользователю будет выведено соответствующее сообщение.

* 1. Выводы по разделу

В ходе работы над дипломным проектом была выполнена установка таких необходимых инструментов для создания проекта, как языки программирования Java и NodeJS, а также установлены дополнительные утилиты, которые необходимы для разработки клиентской и серверной частей проекта.

Была определена общая архитектура проекта, состоящая из трех ключевых частей: база данных, серверная часть, клиентская часть.

Были выбраны архитектуры проектов, такие как «Layered Architecture» для серверной части и модульная для клиентской части с небольшими доработками. Эти разновидности архитектур проектов будут полностью соответствовать поставленным задачам самого дипломного проекта, а также будут удобны и просты для понимания, реализации и дальнейшей поддержки, расширения функционала проекта.

В серверной части приложения были тщательно проработаны и разработаны репозитории, сервисы и контроллеры, были добавлены все необходимые библиотеки, сконфигурирована корректная работа Spring Data Framework для взаимодействия с базой данных MySQL с использованием библиотеки MySqlJavaConnector. Было уделено особое внимание безопасности приложения. Она была сконфигурирована, используя Spring Security. Каждой конечной точке в зависимости от логики была назначена проверка на просто наличие JWT токена при отправке запроса, а также при необходимости осуществляется проверка роли пользователя.

В клиентской части были созданы все необходимые страницы интерфейса, разработана их функциональность, были применены стилевые решения. Также были добавлены все нужные в ходе разработки библиотеки, в частности библиотека Axios для упрощения создания HTTP-запросов при вызове веб-сервиса. Для создания пользовательского интерфейса были задействованы модули React Bootstrap, Bootstrap Icons и другие. Была разработана маршрутизация по приложению и предусмотрена и обеспечена безопасность компонентов приложения.

Таким образом, были настроены и сконфигурированы клиентская и серверная часть дипломного проекта на взаимодействие друг с другом, реализованы основные функциональные возможности и ранее подготовленные алгоритмы приложения.

1. Тестирование программного средства

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 04.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

4 Тестирование программного средства

Лит.

Листов

11

*74218051, 2023*

У

Николайчук А.Н.

Процесс тестирования является не менее важным, чем процесс разработки приложения, поскольку на данном этапе можно найти недочеты в работе приложения и исправить их. В данной главе будут описаны проведения тестирования на серверной и клиентской частях дипломного проекта.

Необходимо протестировать корректность функционирования базовых операций, доступных пользователям. Также стоит удостовериться, что при заведомо неверных действиях система предотвратит их и оповестит об ошибке. Для исключения возможности возникновения ошибок необходимо протестировать каждый из компонентов системы по отдельности.

* 1. Тестирование серверной части

Для тестирования серверной части была использована программа Postman для проверки корректности полученных ответов. На рисунке 4.1 изображен ответ от сервера, а именно получение 401 ошибки, если пользователь ввел неверный пароль при авторизации. В случае если пользователь введет неверный логин, то он получит сообщение, что такого пользователя не существует в системе.

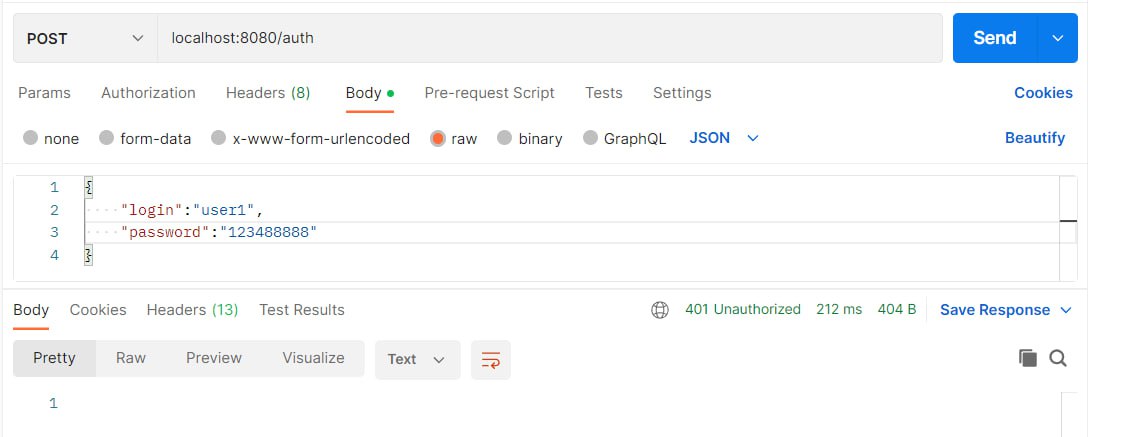


Рисунок 4.1 – Попытка входа с неверным паролем

В случае если пользователь ввел верные данные при авторизации, ему будет отправлен JWT-токен, с помощью которого он получит доступ к возможностям приложения. Теперь пользователь может осуществлять все действия, доступные для его роли. В случае отправления запроса без токена, в ответ будет отправлена статус ошибки 401, который свидетельствует о том, запрос не был применен, поскольку ему не хватает действительных учетных данных для целевого ресурса [18].

Пример успешной авторизации продемонстрирован на рисунке 4.2.

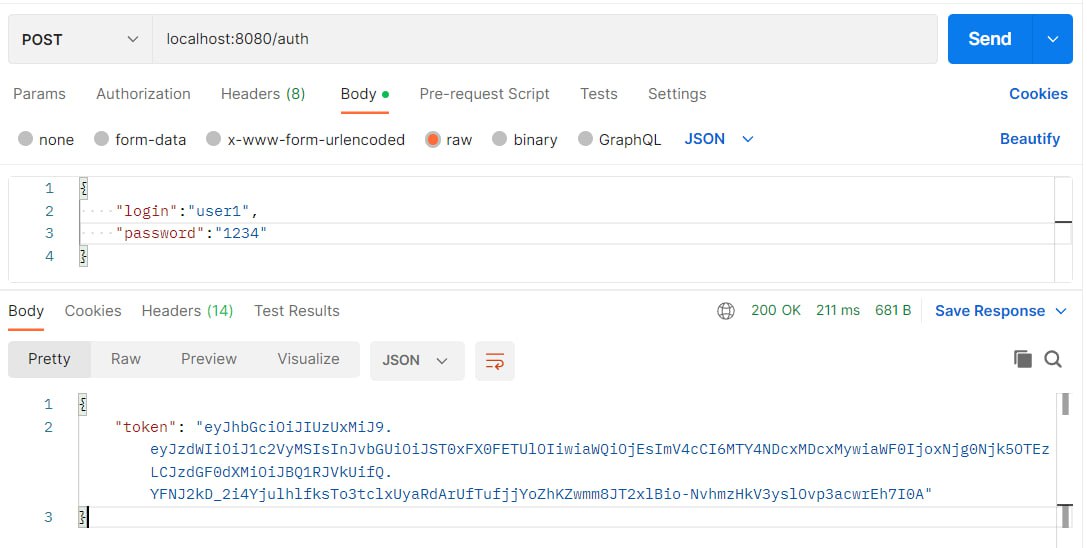


Рисунок 4.2 – Успешная авторизация пользователя

На рисунке 4.3 продемонстрировано получение данных профиля всех сотрудников. Для осуществления авторизации в Postman токен необходимо вставить во вкладке Authorization. Аналогичные проверки были произведены и для других ресурсов, доступ к которым пользователь не должен иметь без статического токена.

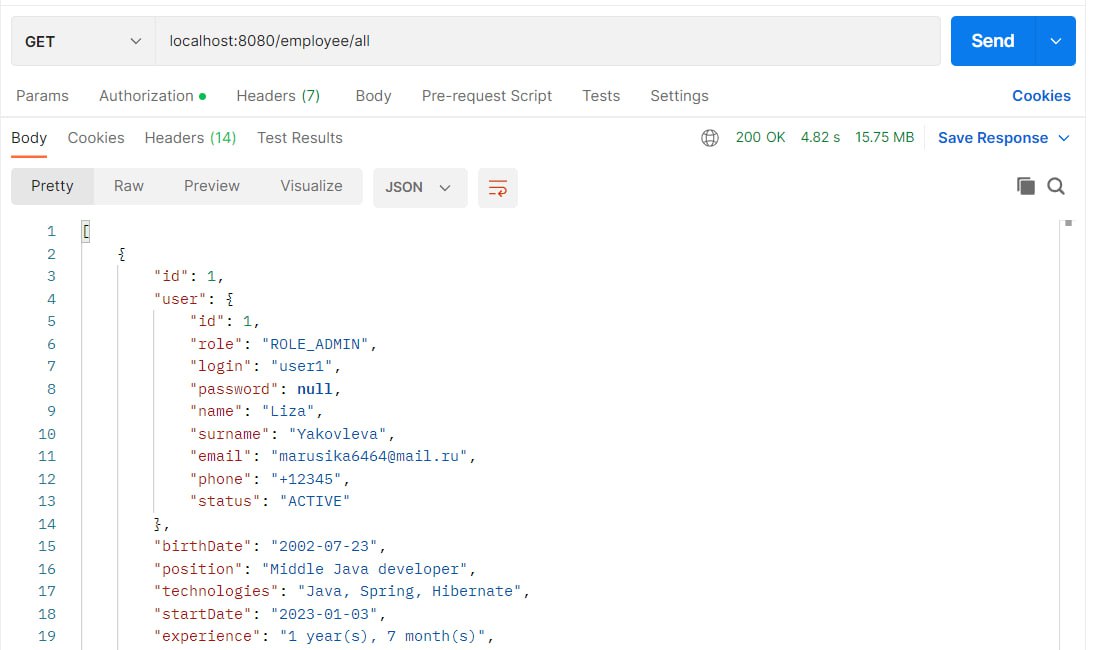


Рисунок 4.3 – Получение данных всех сотрудников

На рисунке 4.4 показана попытка получения данных о всех сотрудниках без прикрепления к запросу токена. Как и ожидалось, получена 401 ошибка. Все запросы к приложению, за исключением авторизационного, нуждаются в наличии токена.

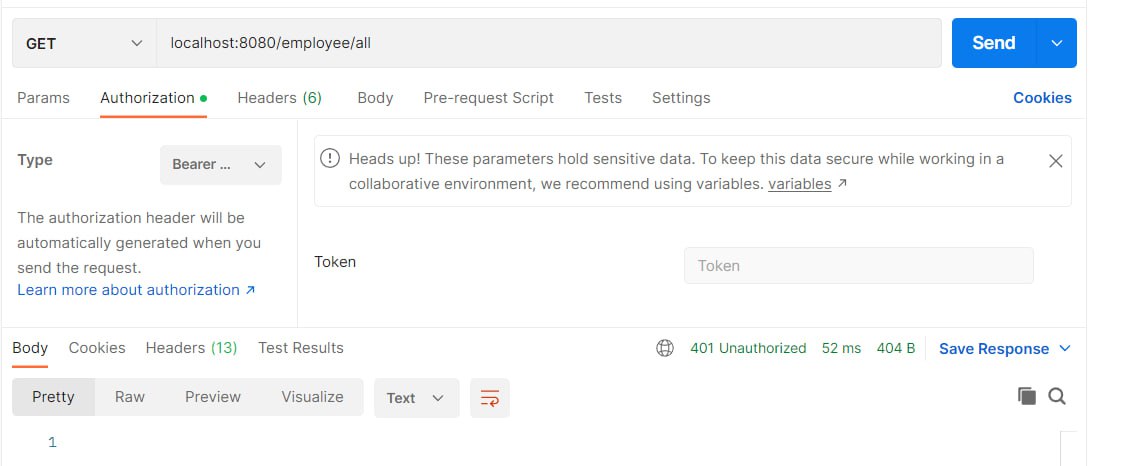


Рисунок 4.4 – Попытка получения данных без токена

Также необходимо удостовериться, что даже при наличии аутентификационного токена, пользователь не сможет успешно выполнить запросы, на которые у него нет прав. Например, на рисунке 4.5 продемонстрировано, как пользователь с ролью сотрудника пытается получить список всех заказов, что доступно только пользователю с ролью менеджера или администратора.

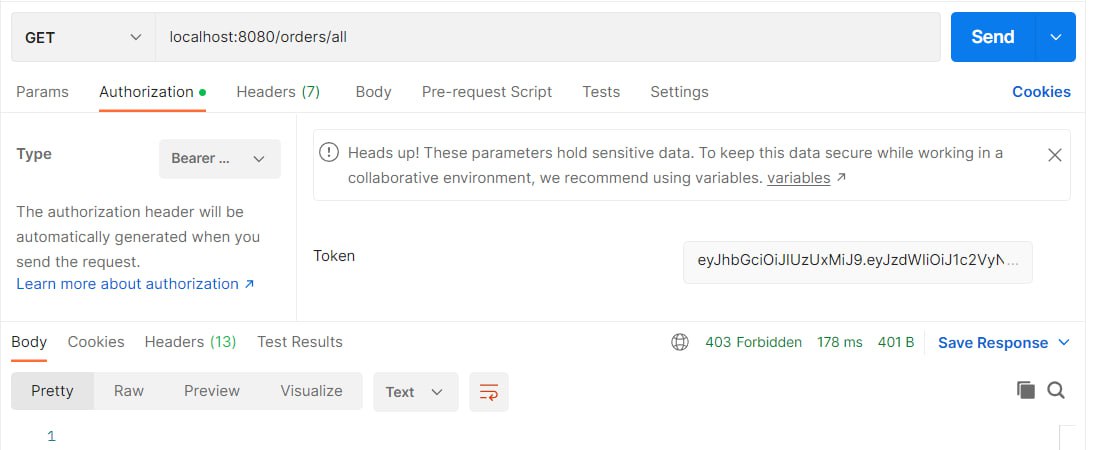


Рисунок 4.5 – Попытка получения данных с недостаточными правами

Как можно увидеть, пользователю была отправлена ошибка со статусом 403, которая означает ограничение или отсутствие доступа к данным. Такая ошибка будет отправлена во всех идентичных ситуациях. Аналогичные проверки были произведены и для других ресурсов, доступ к которым пользователь должен иметь только при наличии соответствующих прав.

Далее необходимо проверить осуществление валидации при POST- и PUT-запросах. Сервер должен вернуть понятное сообщение, чтобы пользователь мог исправить свою ошибку. Например, на рисунке 4.6 приведена попытка редактирования задачи, при которой имя задачи не заполнено. В ответ пользователю ясно сказано, что имя задачи не может быть пустым.

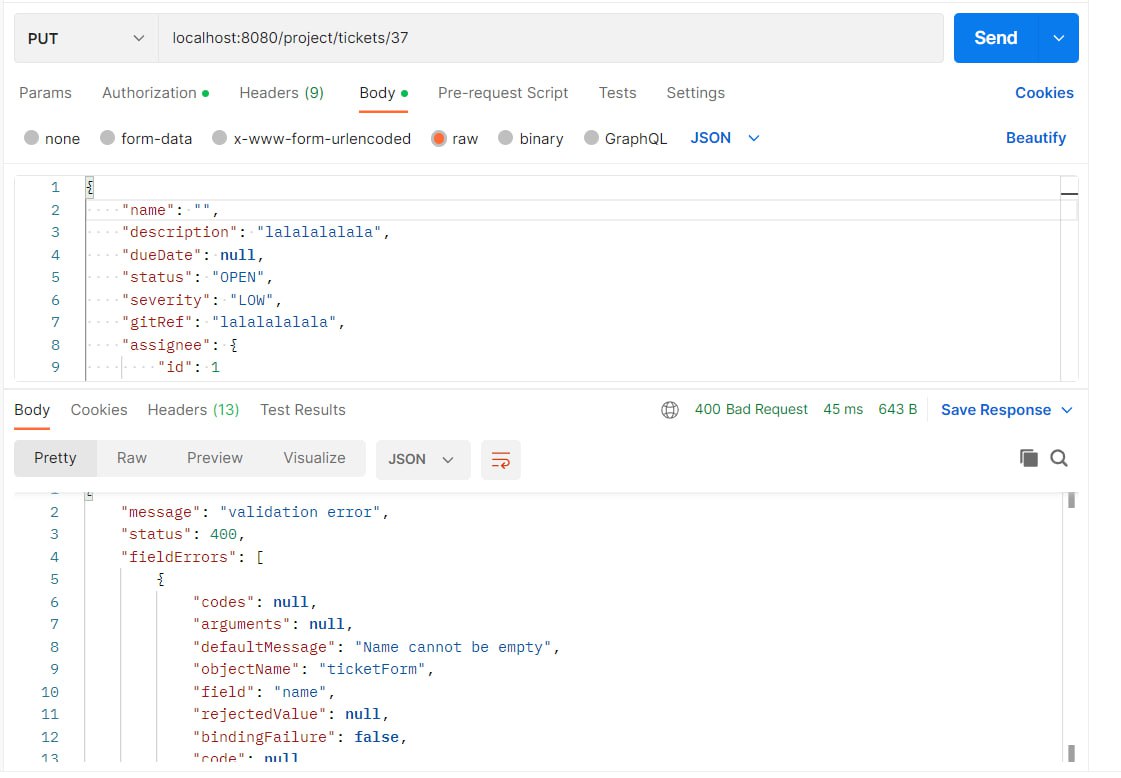


Рисунок 4.6 – Ошибка валидации при редактировании задачи

В случае, если пользователь корректно заполнит все данные и повторит свой запрос, то в ответ будет получен статус 200, информирующий о корректном завершении запроса. Пример приведен на рисунке 4.7.

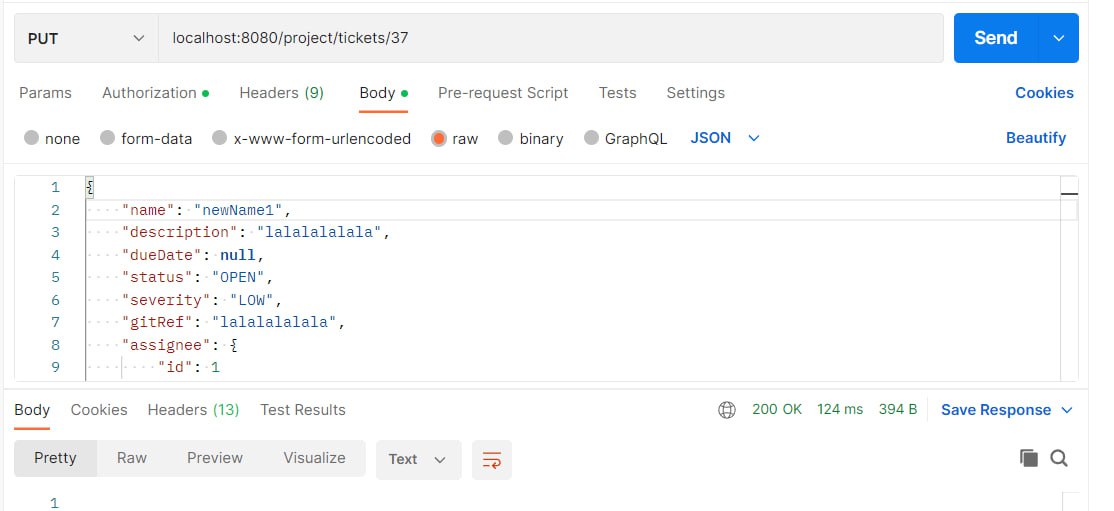


Рисунок 4.7 – Успешное редактирование задачи

Дополнительно проверим успешность редактирования, посмотрев, сохранился ли результат в базе данных. Для этого воспользуемся инструментом для визуализации базы данных MySQL Workbench. На рисунке 4.8 продемонстрировано, что действительно задаче было присвоено новое имя.

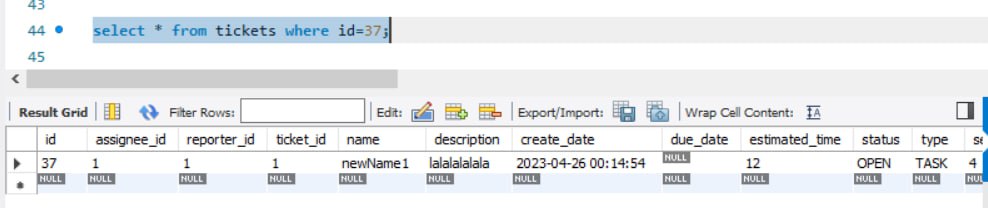


Рисунок 4.8 – Измененная задача в MySQL Workbench

Также важно осуществить проверку на ответ сервера при попытке доступа к ресурсу, которого не существует, поскольку важно, чтобы не возникло исключительной ситуации, приводящей к необработанной ошибке, а было отправлено понятное пользователю сообщение. Например, на рисунке 4.9 происходит попытка получения данных задачи, которой не существует. На это пользователю отображен ответ, информирующий пользователя о данной ситуации.

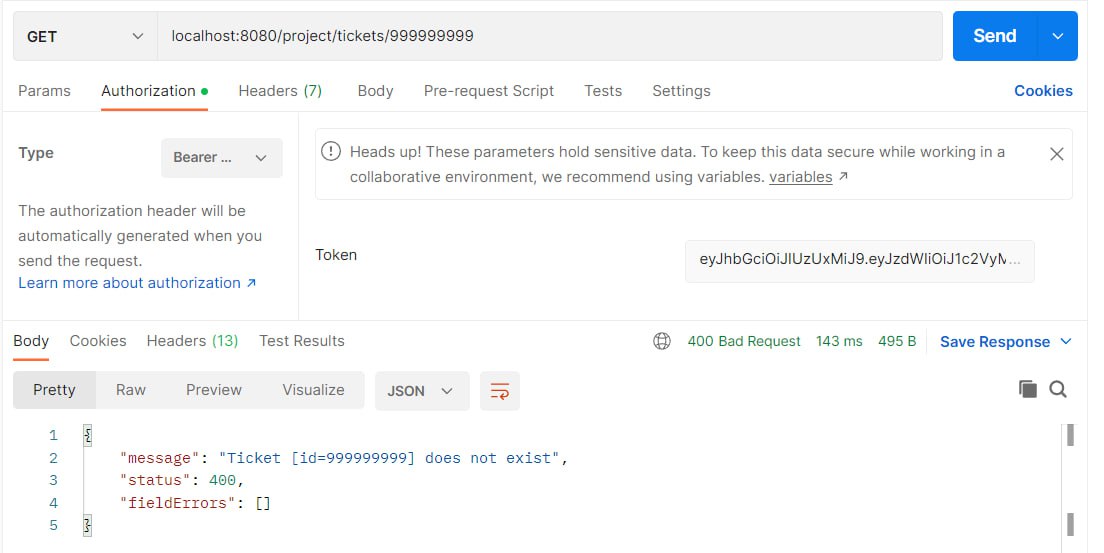


Рисунок 4.9 – Получения данных несуществующего ресурса

Идентичная обработка несуществующего ресурса будет происходит и при попытке получения данных несуществующего пользователя, и при попытке получения данных несуществующей задачи и так далее.

Если аккаунт пользователя по какой-либо причине был заблокирован администратором, даже при вводе правильных логина и пароля заблокированный пользователь ни в коем случае не должен быть допущен в приложение, поэтому важно осуществить проверку данной ситуации. На рисунке 4.10 видно, что при попытке заблокированного пользователя зайти в приложение, была получена ошибка со статусом 423. В данном случае эта ошибка отображает, что аккаунт пользователя был заблокирован и теперь он не имеет доступа к приложению.

В случае если администратор разблокирует пользователя, то ему будет возвращен доступ к приложению, и авторизация пройдет успешно.

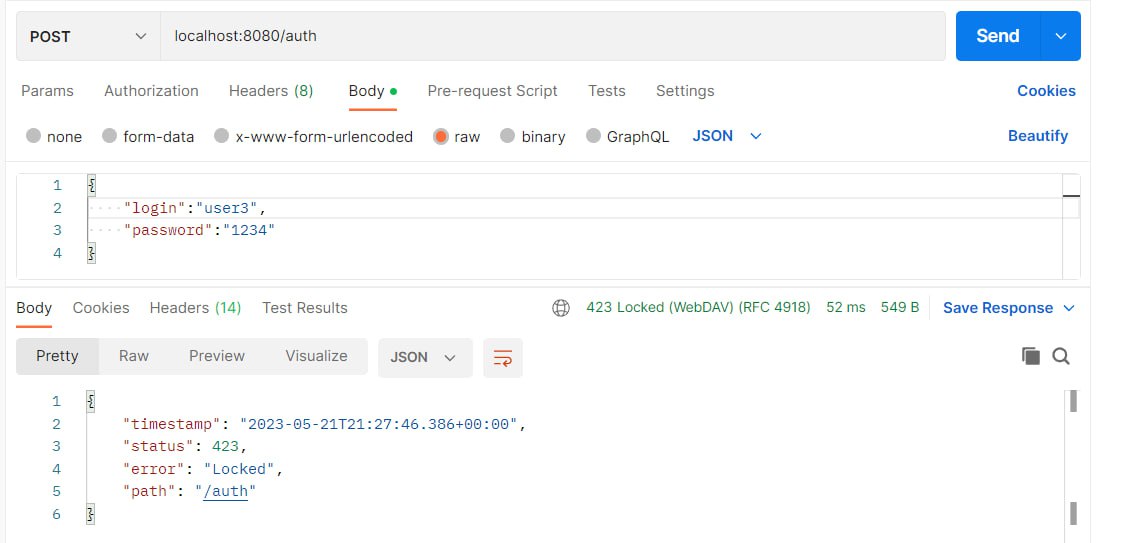


Рисунок 4.10 – Авторизация заблокированного пользователя

Таким образом было проведено ручное тестирование серверной части. Как можно увидеть, все тесты успешно прошли работу, что означает корректную работу модулей всего сервера. Исключительных ситуаций, которые могли привести к нестабильной работе приложения, найдено не было.

* 1. Тестирование клиентской части

Перейдем к тестированию клиентской части приложения и выполним аналогичные действия со стороны клиента и проверим правильность отображаемых данных. Сообщения об ошибках, отправляемые серверной частью, должны корректно отображаться и в клиентском интерфейсе.

Для начала перейдем к странице авторизации. Если пользователь ввел неверный пароль, ему будет отображено сообщение об этом, как показано на рисунке 4.11.

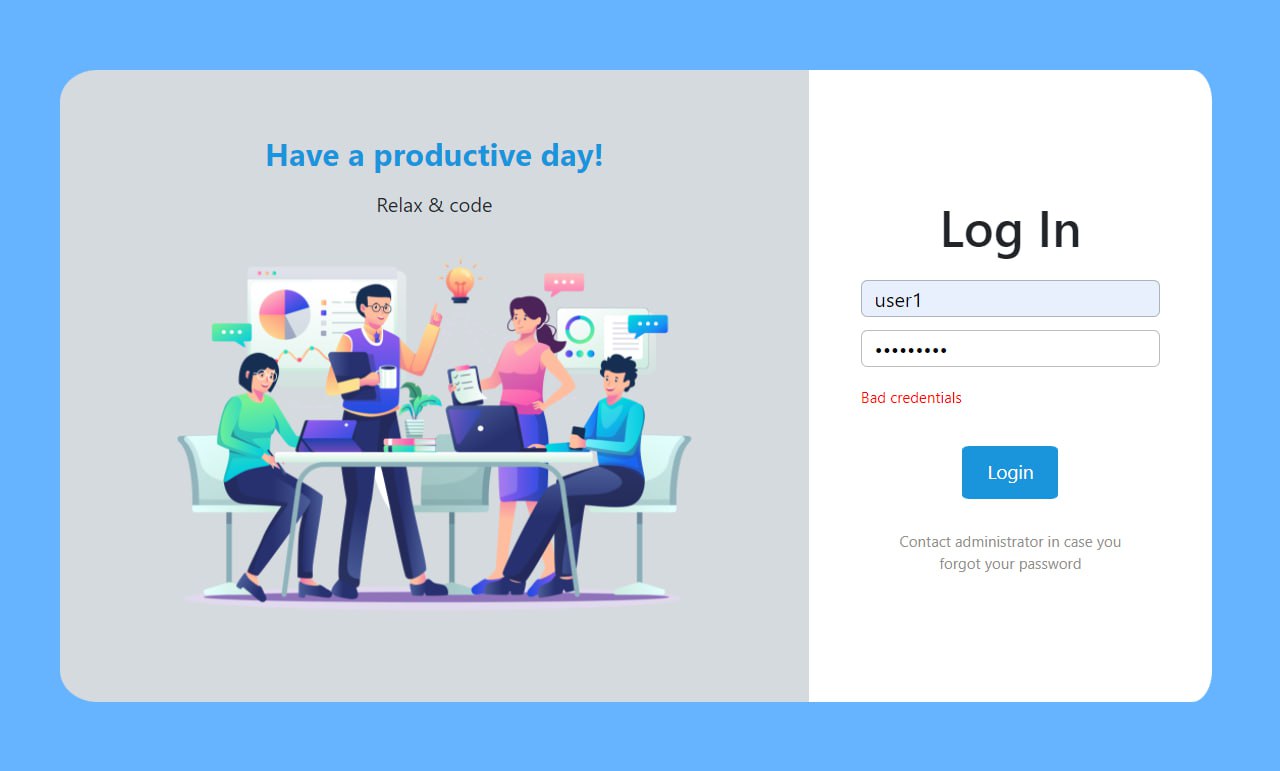


Рисунок 4.11 – Ввод неверного пароля при авторизации

Если заблокированный пользователь совершает попытку авторизации в приложение, то ему будет отображено сообщение об ошибке, как показано на рисунке 4.12.

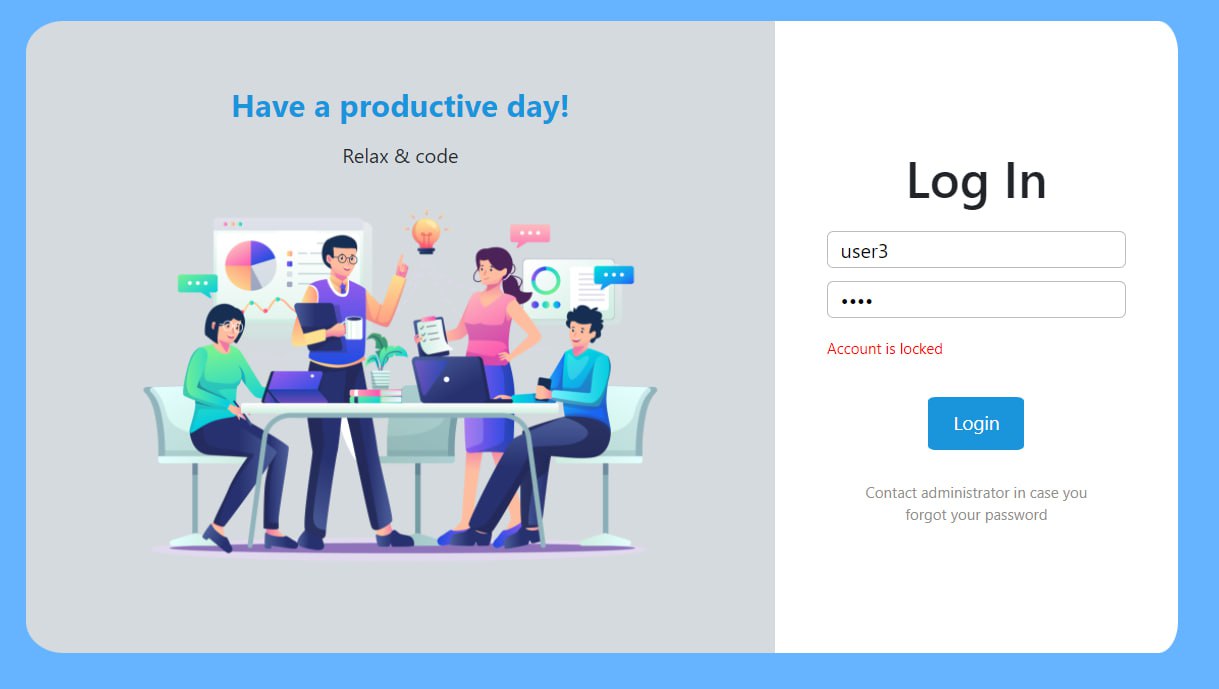


Рисунок 4.12 – Попытка входа заблокированного пользователя

На рисунке 4.13 продемонстрирована попытка создания задачи с некоторыми незаполненными обязательными полями, что нарушает правила валидации. В ответ пользователю отображено сообщение об ошибке, перечисляющее сразу все поля, которые обязательны и не были заполнены при отправке запроса.

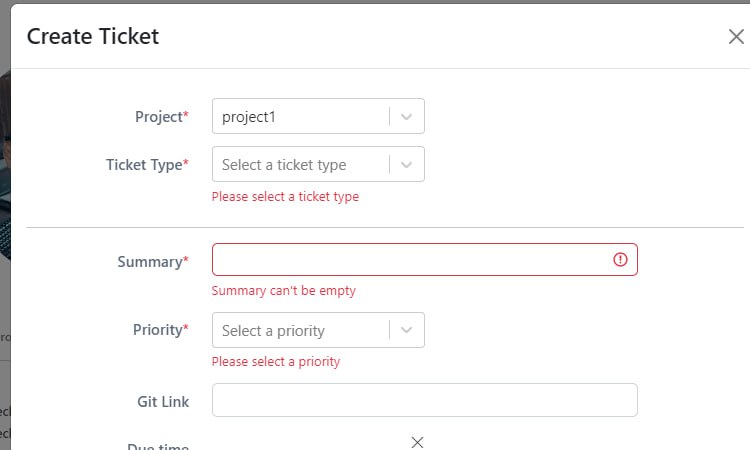


Рисунок 4.13 – Нарушения валидации при создании задачи

Если сотрудник попытается внести изменения на дэшборде чужого проекта, то появится сообщение, уведомляющее о запрете на данное действие, как показано на рисунке 4.14, и внесенные изменения будут отменены. Так же будет и в случае попытки клиента перетащить задачу на дэшборде даже своего проекта.

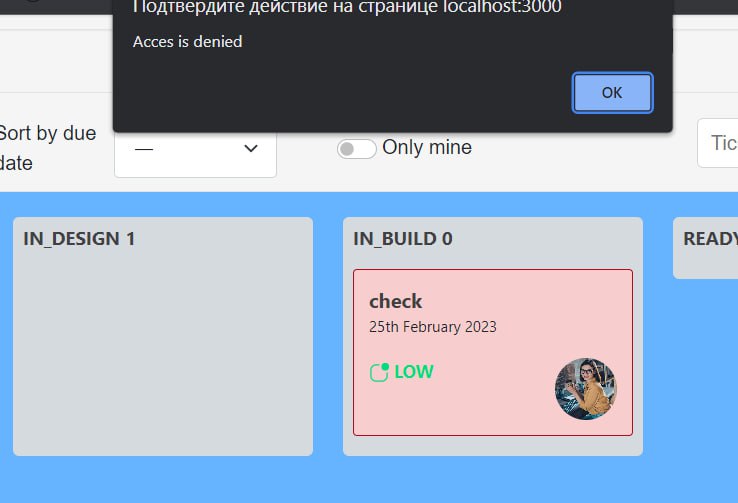


Рисунок 4.14 – Попытка изменения чужого дэшборда

Также было проверено, что при попытке сотрудника или клиента попасть на страницы, доступные только менеджерам и администратору, все работает корректно, и они действительно будут перенаправлены на доступные их ролям страницы.

Также была протестирована безопасность при использовании куков. Если, например, клиент путем изменения в куках номера проекта попытается получить доступ к дэшборду чужого проекта, он сразу будет переадресован на страницу с дэшбордом его проекта, а куки также автоматически будут заменены на корректные. Такие же операции будут произведены при попытке клиента получить доступ к детальной информации чужого проекта или задачи.

На каждой странице для каждой роли было проверено, что содержатся только те функции и действия, на которые у данной роли есть права. В частности, на рисунке 4.15 видно, что сотрудник может изменять и удалять только личные комментарии.



Рисунок 4.15 – Комментарии к задаче

Когда администратор решает деактивировать или заблокировать пользователя, для подтверждения действия ему отображается соответствующее подтверждающее всплывающее окно, как показано на рисунке 4.16.

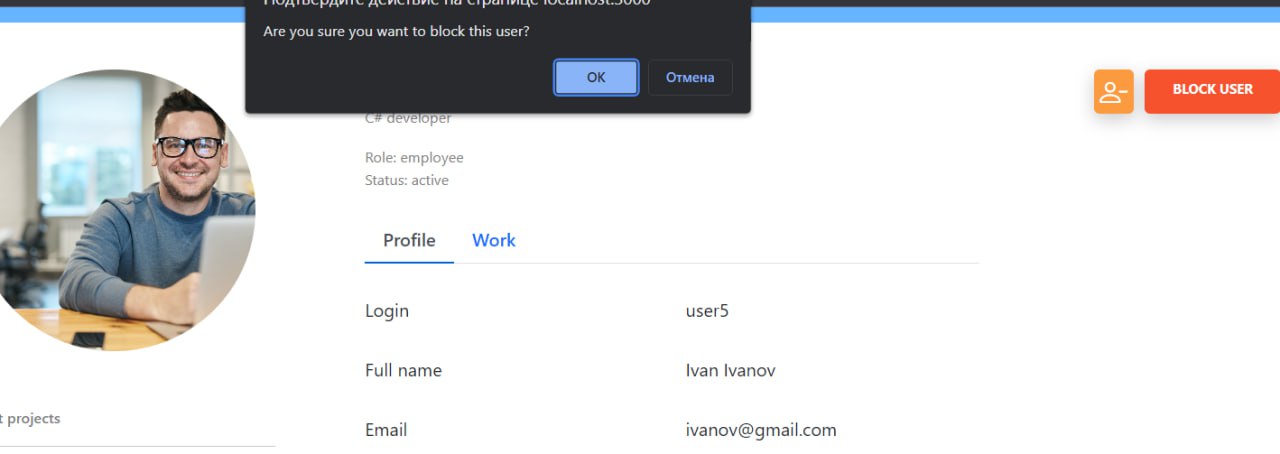


Рисунок 4.16 – Подтверждающее всплывающее окно

После того, как администратор деактивировал пользователя, данному пользователю на электронную почту приходит письмо с новым временным паролем для авторизации. Пример приведен на рисунке 4.17.

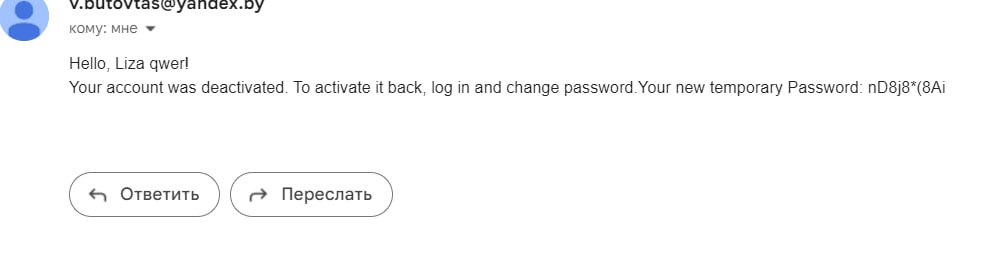


Рисунок 4.17 – Электронное письмо для деактивированного пользователя

Теперь при первом после деактивации входе этого пользователя в приложение, ему будет отображено модальное окно для смены пароля. Такое окно продемонстрировано на рисунке 4.18. В нем необходимо ввести текущий временный пароль, а также новый и повторить новый пароль. Если новый пароль в двух полях не совпадает, то будет отображена ошибка, а кнопка подтверждения будет недоступной.

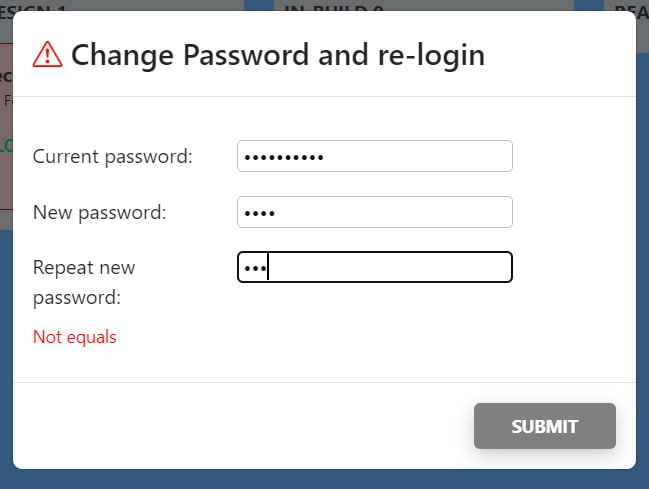


Рисунок 4.18 – Модальное окно для смены пароля

При редактировании задачи было проверена возможность перетаскивания файла в область для прикрепленного файла с целью замены. Как можно увидеть на рисунке 4.19, данная функция работает исправно.

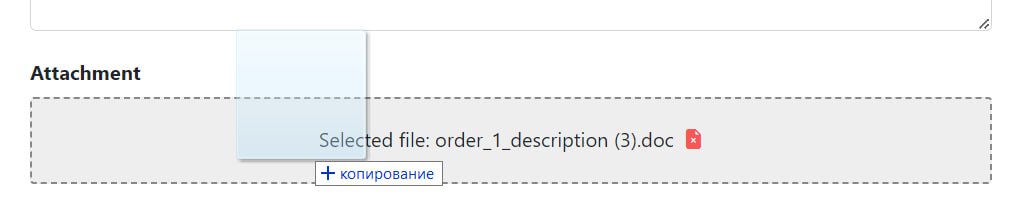


Рисунок 4.19 – Замена прикрепленного к задаче файла

На рисунке 4.20 показано, что при нажатии на файл, он был успешно скачан в папку загрузки на локальную машину.

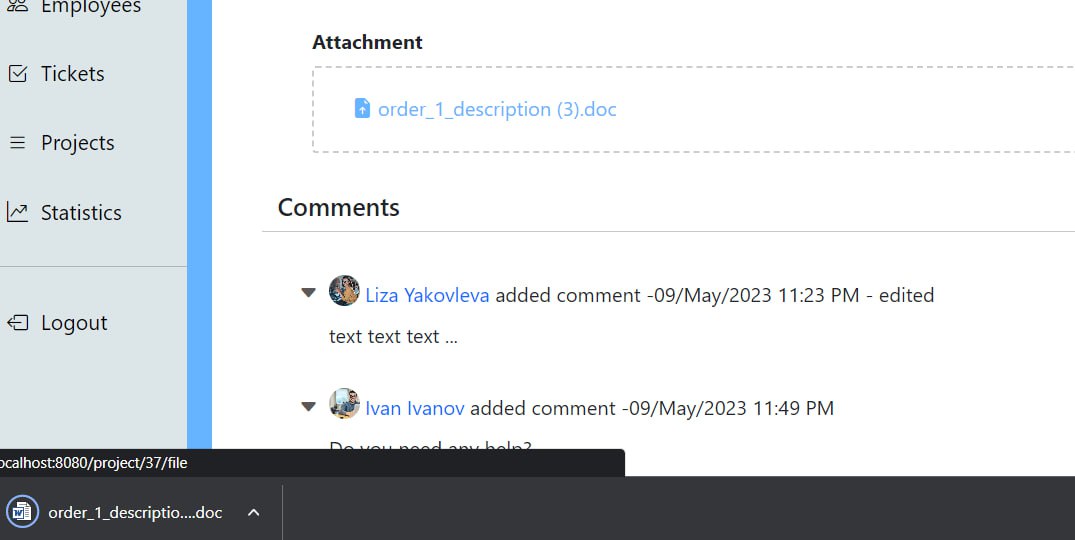


Рисунок 4.20 – Скачивание файла

Весь интерфейс приложения написан на английского языке. Это объясняется тем, что делается расчет на то, что все пользователи владеют английским языком, поскольку в настоящее время это одно из самых важных требований в IT-сфере. А также важным фактором является возможность работы с клиентами со всего мира за счет отсутствия языкового барьера в приложении. Но тем не менее, хоть в приложении и нет встроенной смены языка, при необходимости пользователь может перевести любую страницу приложения, воспользовавшись встроенным в браузер инструментом, таким как Google Translate, благодаря которому приложение может быть переведено буквально на любой язык мира. Пример перевода страницы профиля сотрудника с использованием данного средства представлен на рисунке 4.21.

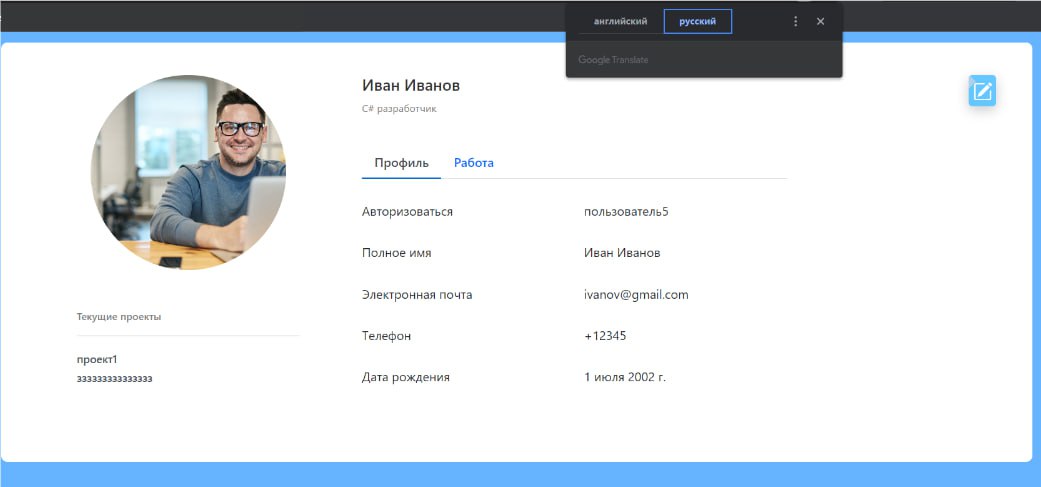


Рисунок 4.21 – Перевод страницы с использованием Google Translate

Тестирование клиентской части приложения можно считать успешно завершенным. Проблем выявлено не было.

* 1. Выводы по разделу

Основной целью данного раздела являлось тестирование всего дипломного для гарантии его целостной и стабильной работы. В качестве варианта проведения тестирования было выбрано ручное тестирование. Оно позволяет точно проверить отдельные элементы всего приложения в различных ситуациях. Были проверены как «позитивные», так и «негативные» тестовые случаи.

Для тестирования серверной части использовалось приложение Postman, которое позволяет формировать запросы, наполнять их данными и отправлять на сторону сервера. В ходе тестирования сервера были проверены такие основные ситуации, как получение данных без авторизации, отправка невалидных данных, проверка прав на доступ к данным. В ходе тестирования были проверены такие действия, как: авторизация, получение данных о всех сотрудниках, редактирование задачи.

Для тестирования клиентской части проекта был использован браузер Google Chrome. В ходе тестирования было проверено поведение клиентской части при пропуске необходимых полей, проверена работоспособность автоматической отправки электронных писем. На стороне клиента были протестированы такие действия как: авторизация, перетаскивание задач на дэшборде, редактирование файла задачи, деактивация пользователя, перевод страниц.

В ходе проведения тестирования всего проекта были обнаружены и устранены все уязвимости, которые могли повлечь за собой нарушение стабильности работы всего приложения и привести к потере пользовательских данных.

1. Руководство пользователя

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 05.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

5 Руководство пользователя

Лит.

Листов

14

*74218051, 2023*

У

Николайчук А.Н.

* 1. Развертывание приложения в среде Docker

Для проверки работоспособности приложения прежде всего необходимо установить на компьютере Docker. Никаких дополнительных экземпляров базы данных или, например, платформ как Java, NodeJS устанавливать не требуется, так как все образы будут скачаны через зависимости в конфигурационных файлах Docker из общедоступного централизованного хранилища. Таким образом, проект не требует жесткой привязки к какой-либо системе и может быстро и легко разворачиваться на любой операционной системе. Для создания самих контейнеров Docker нужно выполнить команду «docker compose up» в терминале, открытом в директории, где находится файл docker-compose. После этого все указанные в файле контейнеры будут автоматически созданы и запущены, и можно начать работать с приложением. Для этого нужно сделать переход по следующему URL в браузере: «http://localhost:3000».

Пример опубликованного в Docker с использованием Docker Compose приложения выглядит следующим образом, как показано ниже на рисунке 5.1.

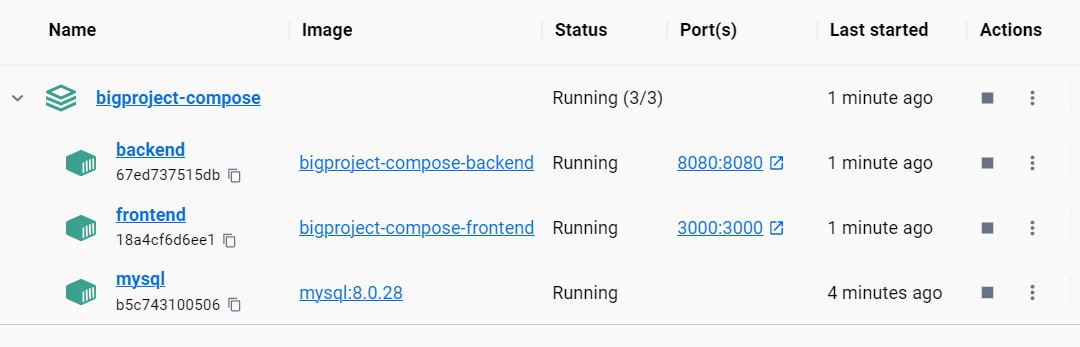


Рисунок 5.1 – Развернутое на Docker приложение

Чтобы проверить статус контейнеров можно выполнить команду как на рисунке 5.2, которая отобразит все существующие контейнеры и их состояния.

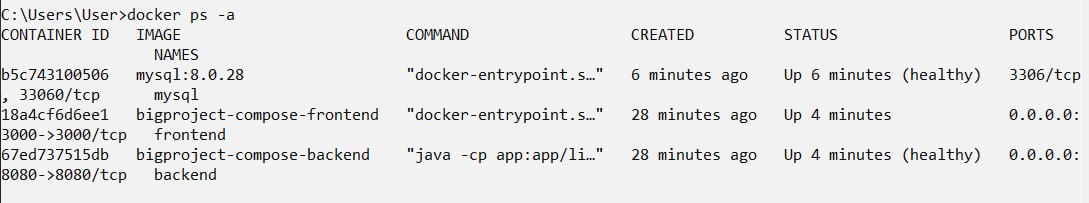


Рисунок 5.2 – Просмотр всех контейнеров в Docker

В данном случае запуск происходил на основе операционной системы Windows. Чтобы запустить разработанное приложение на другой машине, необходимо либо скопировать на нее все Docker-файлы, включая docker-compose, а самое главное код приложения, либо после создания образов, можно загрузить их в облачное хранилище, так называемое docker registry. В случае второго варианта будет возможность запустить приложение где угодно и когда угодно, не перенося на машину само приложение и конфигурационные файлы. Конечно, для запуска контейнеров необходим docker-compose файл, но имея готовые образы, его можно быстро и легко написать самостоятельно. При загрузке образов в репозиторий хранилища важно сделать репозиторий приватным, чтобы только определенные лица имели к нему доступ. Однако, чтобы иметь возможность создавать более одного приватного репозитория, будет необходимо приобрести подписку.

Образ приложения, загруженного в docker registry представлен на рисунке 5.3. В данном случае используется DockerHub. Как можно увидеть, один из образов был сделан приватным в целях безопасности.

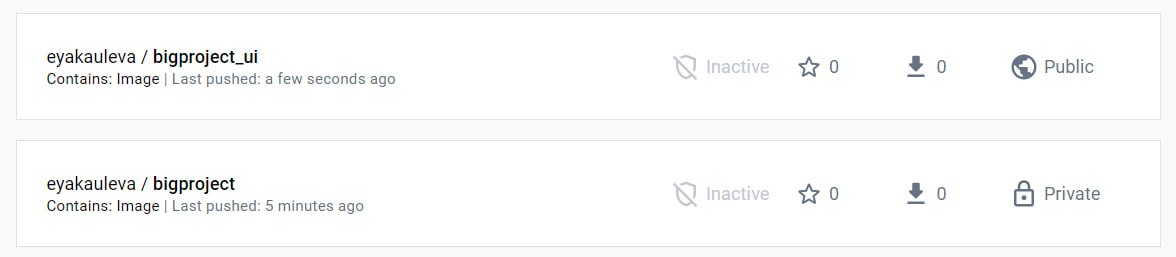


Рисунок 5.3 – Загруженные в docker registry образы приложения

Теперь для демонстрации можно легко запустить весь проект на другой машине с иной операционной системой, например, MacOS. Для этого будет использован идентичный ранее использованному docker-compose файл, за исключением того, что в данном случае вместо указания путей к Dockerfile файлам, будет указано полное название образов из хранилища, как это уже сделано в контейнере для базы данных MySQL, который использует готовый образ из хранилища. После запуска файла, будет получен результат, представленный на рисунке 5.4. Как можно увидеть, все контейнеры были успешно созданы и запущены.

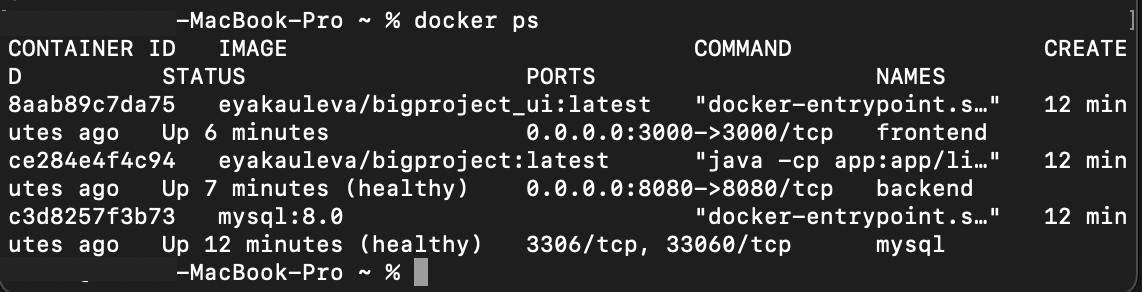


Рисунок 5.4 – Развернутое на Docker приложение на ОС MacOS

Таким образом, любая операционная система, имеющая поддержку контейнеризации или предустановленное средство контейнеризации Docker, может без всяких проблем запускать данное приложение и выполнять его стабильно без надобности установки дополнительных инструментов.

* 1. Руководство для сотрудника

Роль сотрудник предназначена для пользователя, являющегося сотрудником компании без особых прав. У него есть доступ ко всем базовым возможностям.

Для того чтобы начать работу с приложением, необходимо авторизоваться путем ввода своих логина и пароля. Страница авторизации приведена на рисунке 5.5.

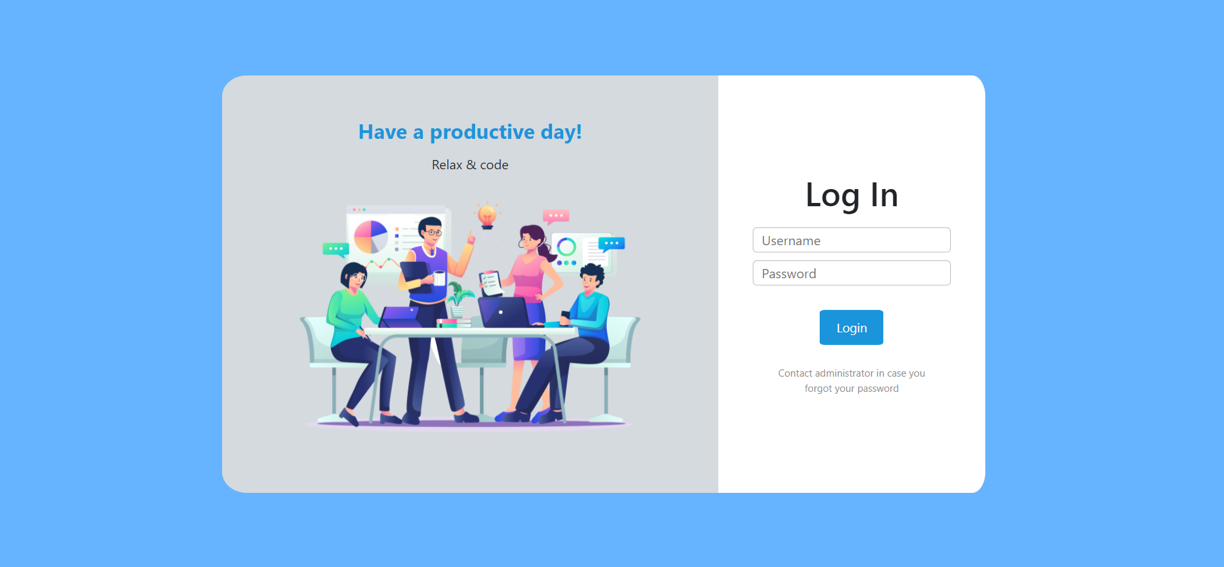


Рисунок 5.5 – Страница авторизации

После прохождения авторизации любой пользователь-сотрудник будет перенаправлен на страницу с его профилем. Пример страницы пользователя представлен на рисунке 5.6. Здесь во вкладке Profile отображена общая информация о пользователе, а во вкладке Work данные о профессиональной деятельности сотрудника. Пользователь может редактировать данные, нажав на кнопку редактирования, находящуюся в правом верхнем углу. Под фотографией находится список проектов, к которым закреплен сотрудник и которые находятся в разработке.

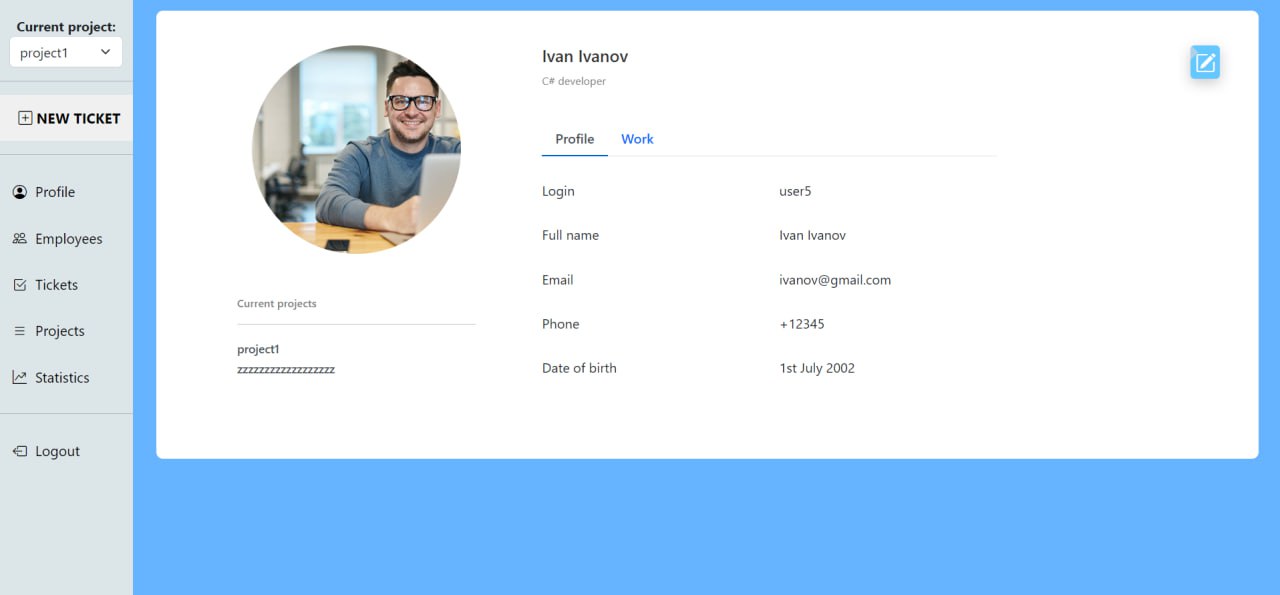


Рисунок 5.6 – Страница пользователя

Навигацию по приложению можно осуществлять, используя меню, находящееся с левой стороны. Вверху меню расположен выпадающий список, где сотрудник может переключать проекта, с которым он работает в данный момент. Ниже находится кнопка для создания задачи. Нажав на нее, появится модальное окно для создания задачи, представленное на рисунке 5.7. Поля, помеченные звездочкой, обязательны к заполнению. По умолчанию в поле Project будет выбран проект, выбранный как текущий в меню, но его можно изменить, нажав на поле, после чего будет отображен выдающий список с проектами для выбора. Для прикрепления файла к задаче можно либо кликнуть по полю Attachment, либо перетащить файл прямо в поле.

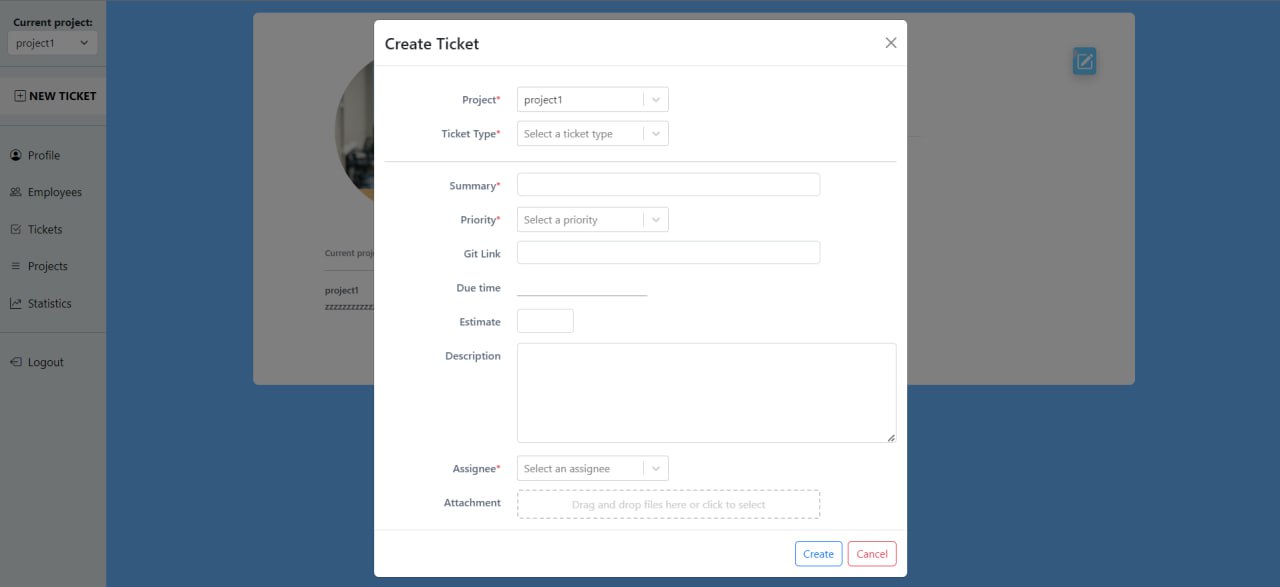


Рисунок 5.7 – Модальное окно для создания задачи

Далее в меню находится клавиша Profile, при нажатии направляющая пользователя на страницу с его профилем как на рисунке 5.6. Кнопка Employees в меню отображает пользователю всех сотрудников компании. Данная страница представлена на рисунке 5.8. Можно осуществлять поиск по имени и/или фамилии, используя поисковое поле в правом верхнем углу.

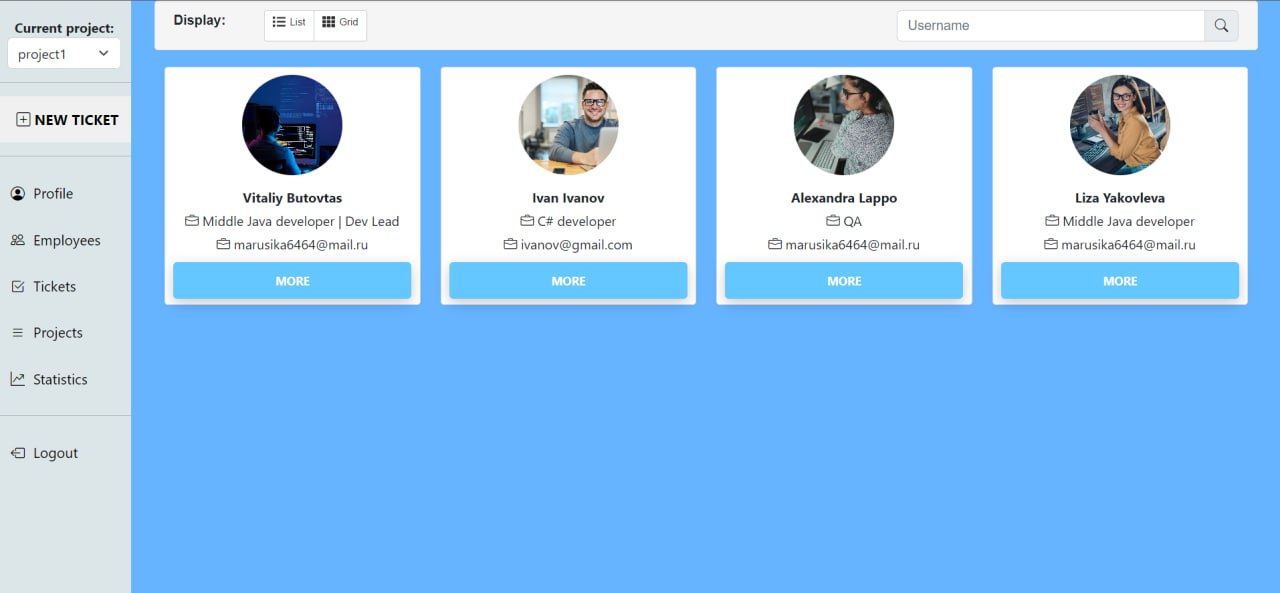


Рисунок 5.8 – Страница со всеми сотрудниками

Также есть возможность представления в виде сетки, как на рисунке 5.8, и в виде раскрывающегося списка, как на рисунке 5.9. Чтобы перейти на профиль сотрудника необходимо нажать на кнопку More.

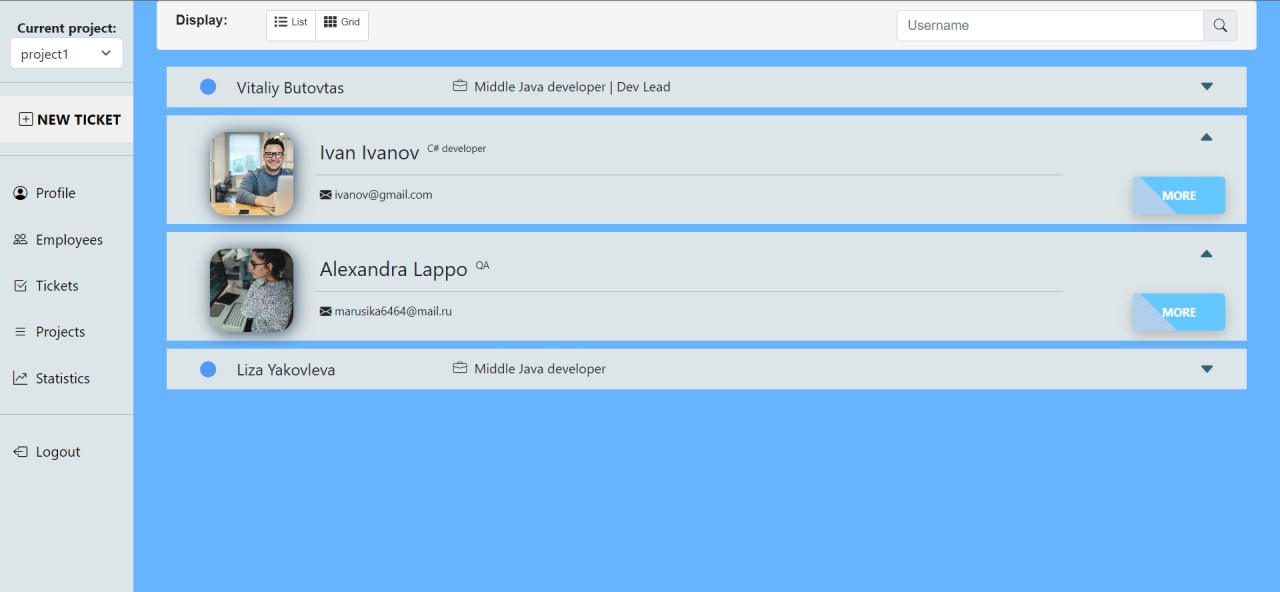


Рисунок 5.9 – Страница со всеми сотрудниками в виде списка

Следующая клавиша меню это Tickets. При нажатии на нее пользователь будет перенаправлен на страницу с дэшбордом, представленную на рисунке 5.10. Дэшборд на ней будет представлен для проекта, выбранного в меню в качестве текущего. При изменении текущего проекта в меню дэшборд будет перезагружен. В самом верху странице слева дополнительно указано имя проекта дэшборда.

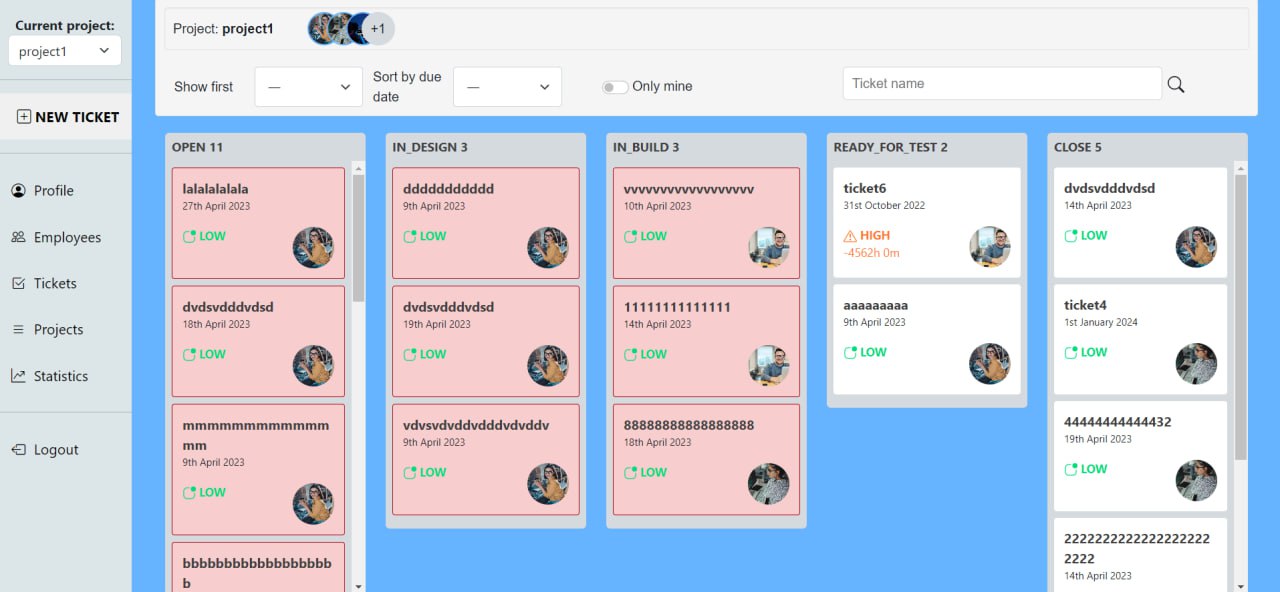


Рисунок 5.10 – Страница с дэшбордом

Справа от названия проекта изображены иконки с фотографиями нескольких сотрудников, которые работают на проекте. При нажатии на них откроет модальное окно с полноценным списком всех сотрудников, занятых на проекте. По списку можно осуществлять поиск по имени и/или фамилии. Пример такого окна будет представлен далее при рассмотрении страницы проекта.

На панель ниже представлены инструменты для произведения поиска, фильтрации и сортировки по задачам. Поиск осуществляется на основе названия задачи, с помощью фильтрация можно отобразить только те задачи, которые закреплены за вами, а сортировку можно выполнить либо по приоритету задачи, тогда задачи с выбранным статусом будут отображены первыми, либо по сроку выполнения задачи по убыванию или по возрастанию. Также в полях сортировки есть возможность вернуть порядок задачи к исходному, выбрав в них пустое значение.

В центре страницы расположены сами задачи, разделенные по их статусам. Красным цветом отмечены задачи, в которых просрочен срок выполнения. Задачи можно перемещать курсором внутри колонки, а также в другие колонки. Чтобы перейти на страницу задачи нажмите на название задачи. Пример страницы для конкретной задачи представлен на рисунке 5.11.

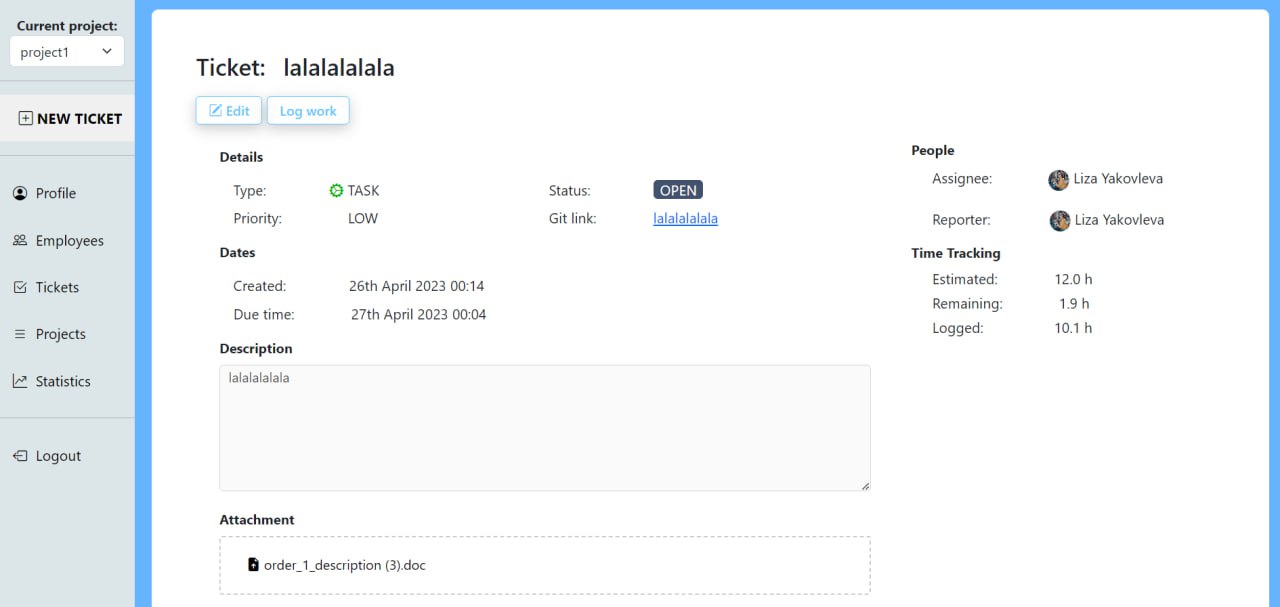


Рисунок 5.11 – Страница задачи

На страницы представлена вся детальная информация о задаче. Редактировать задачу может любой сотрудник, который работает на проекте, которому принадлежит задача, а также любой администратор. Чтобы отметить время, затраченное на выполнение задачи, то есть залогировать его, необходимо нажать на соответствующую кнопку на страницу, после чего появится модальное окно, представленное на рисунке 5.12. Здесь можно указать количество часов, потраченных на выполнение задачи в текущий день, а также изменить оценочное время на выполнение задачи. Если оценочное время будет меньше, чем уже затраченное, то на странице задачи в графе с оценочным временем будет указано затраченное время. Логирование времени могут выполнять те же лица, что и изменение задачи.

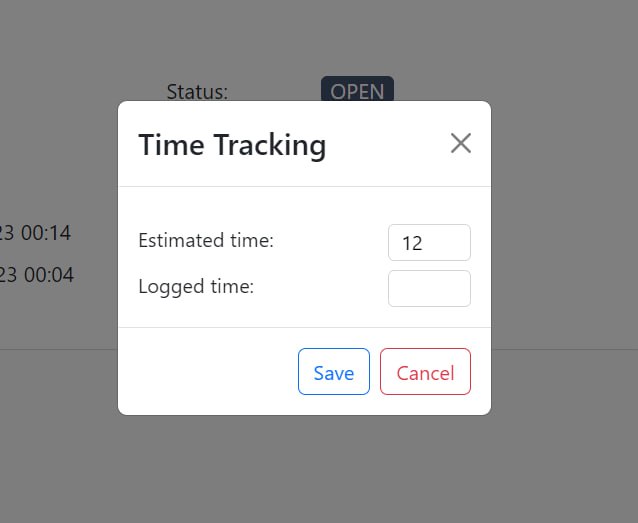


Рисунок 5.12 – Окно для логирования времени

Также под задачей можно оставлять комментарии. Они все также находятся на странице с задачей в самом ее низу и представлены на рисунке 5.13. Есть возможность сворачивать комментарии. Оставлять комментарии под задачей может любой сотрудник. Пользователи могут редактировать и удалять свои комментарии, а администратор может редактировать и удалять любые комментарии.

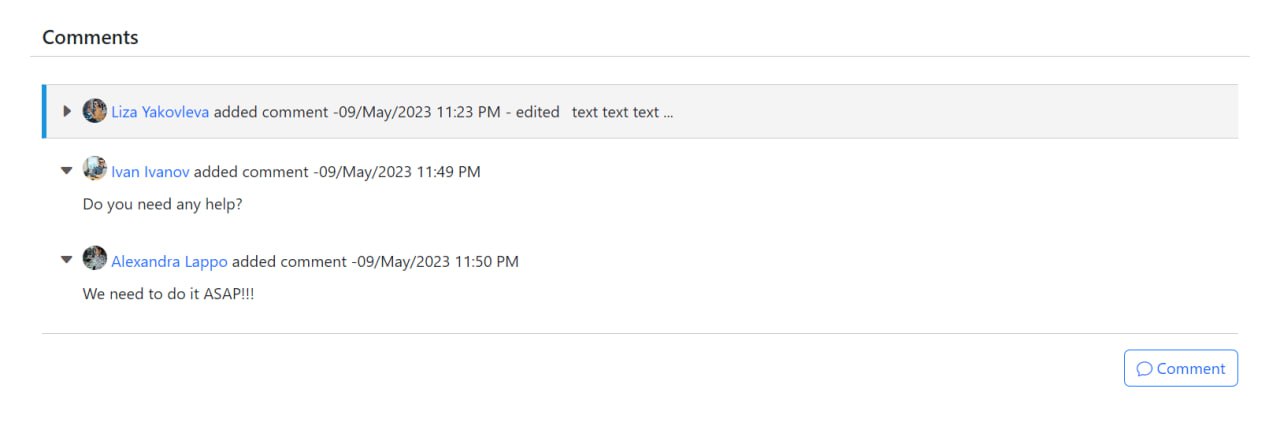


Рисунок 5.13 – Комментарии к задаче

Далее рассмотрим страницу с проектами. Она изображена на рисунке 5.14. На ней представлены все существующие проекты компании.

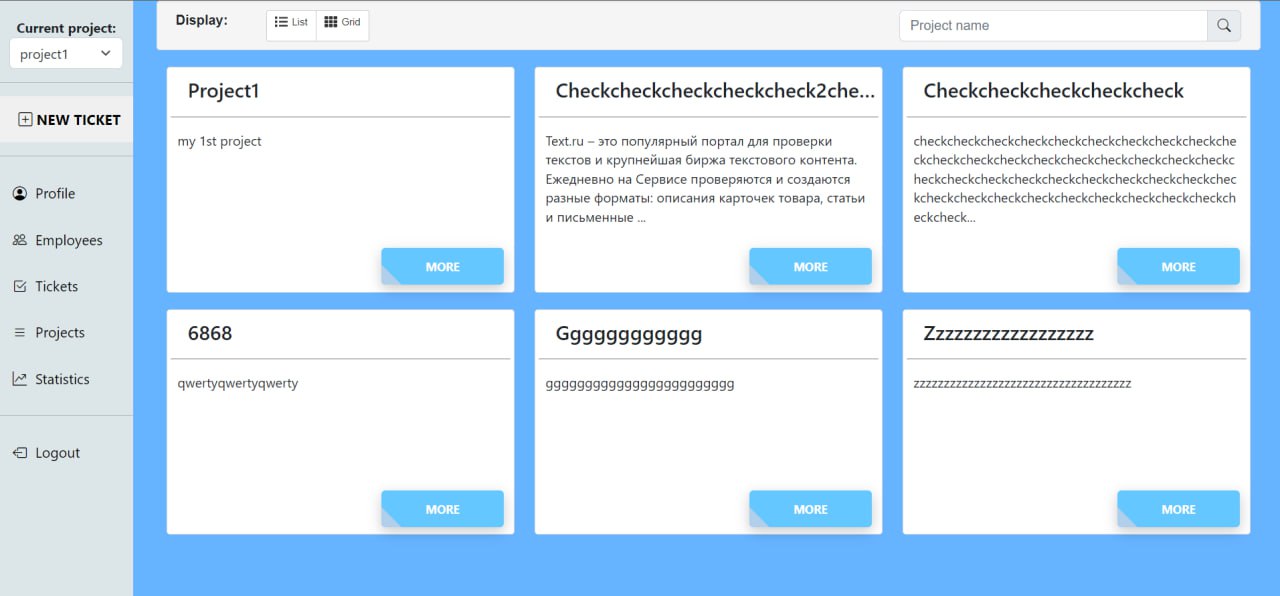


Рисунок 5.14 – Страницы со всеми проектами

Можно отображать данные в виде сетки или списка, а также осуществлять поиск по проектам по названию. Чтобы перейти на страницу проекта необходимо нажать на кнопку More. Пример страницы проекта представлен на рисунке 5.15. В целом она выглядит идентично странице задачи, за исключением того, что тут отсутствует возможность логировать время и внизу страницы отображены все задачи, относящиеся к данному проекту.

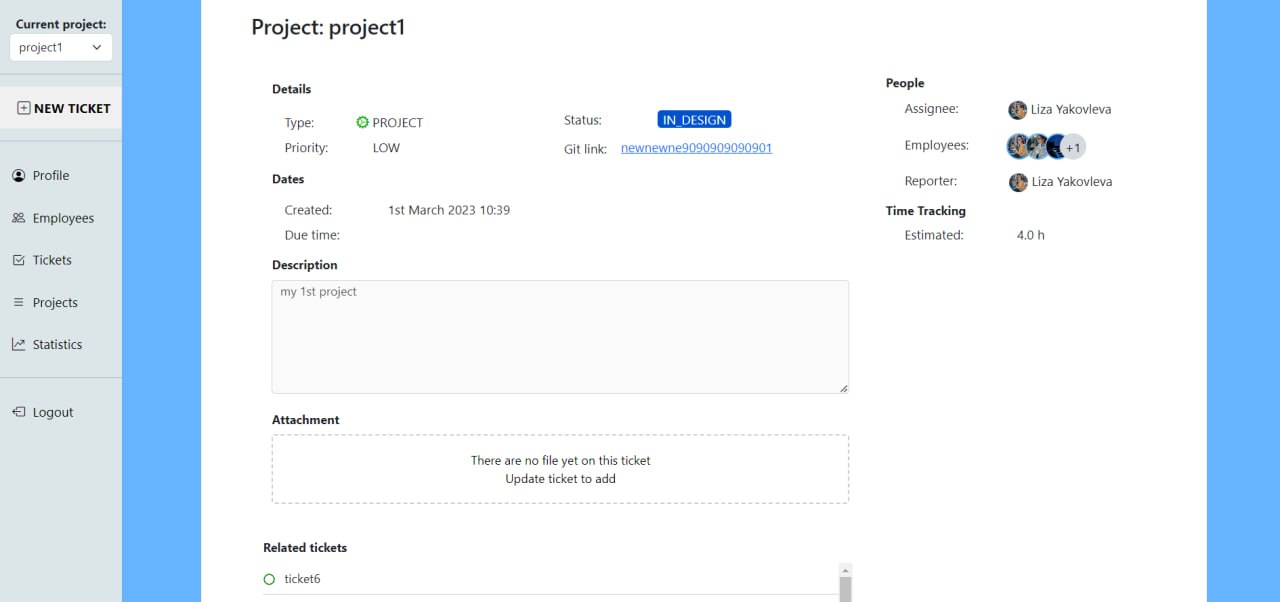


Рисунок 5.15 – Страница проекта

В правой части экрана отображено поле, отображающее сотрудников, работающих на данном проекте. При нажатии на него будет отображено модальное окно, содержащее уже полный список участников проекта, где можно осуществлять среди них поиск по имени и/или фамилии. Пример этого окна представлен на рисунке 5.16.

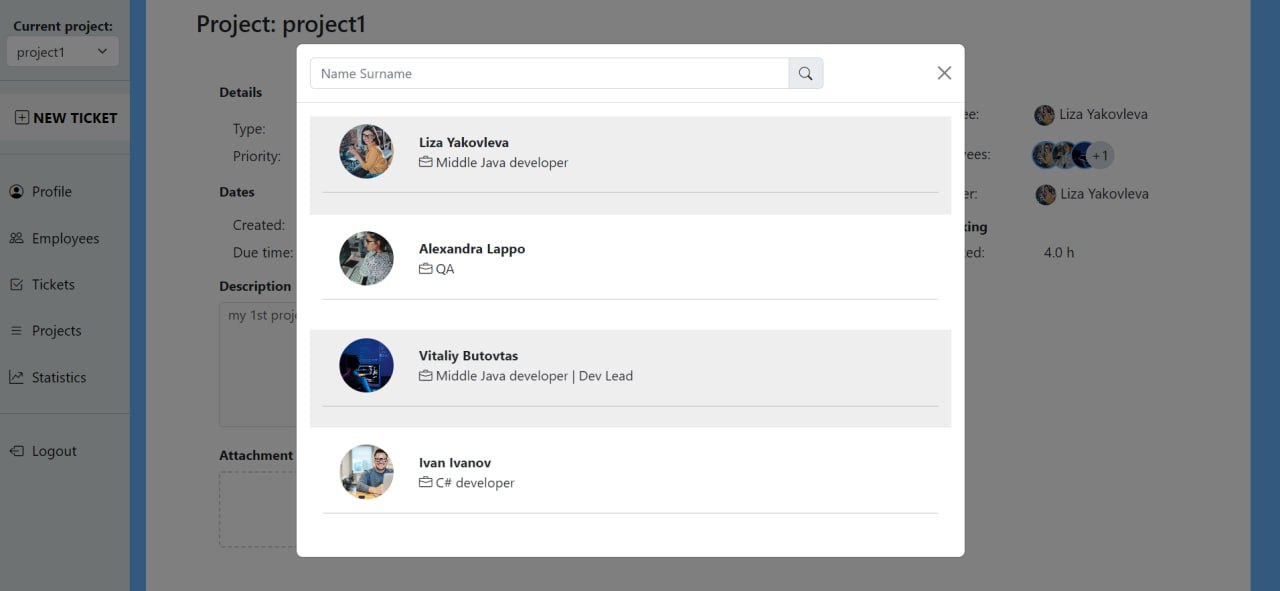


Рисунок 5.16 – Модальное окно со всеми сотрудниками проекта

Следующий пункт навигационного меню – это страница со статистикой. Для сотрудников здесь представлена как личная статистика, пример которой продемонстрирован на рисунке 5.17, так и отдельная статистика по проектам.

На странице с личной статистикой приведены следующие показатели. Слева изображена столбчатая диаграмма, отображающая количество залогированных пользователем часов по дням. Для переключения между неделями используются стрелки над диаграммой. Справа отображено сколько у пользователя на данный момент незакрытых задач, а также количество задач, у которых превышен установленный срок выполнения. Цвет данных кружков зависит от количества задач. В лучшем случае они будут зеленого цвета, в случае средней – ситуации желтого, а в большее худших вариантах – красного цвета. Под ними находится список задач, у которых вышло время завершения или закончится сегодня или завтра.

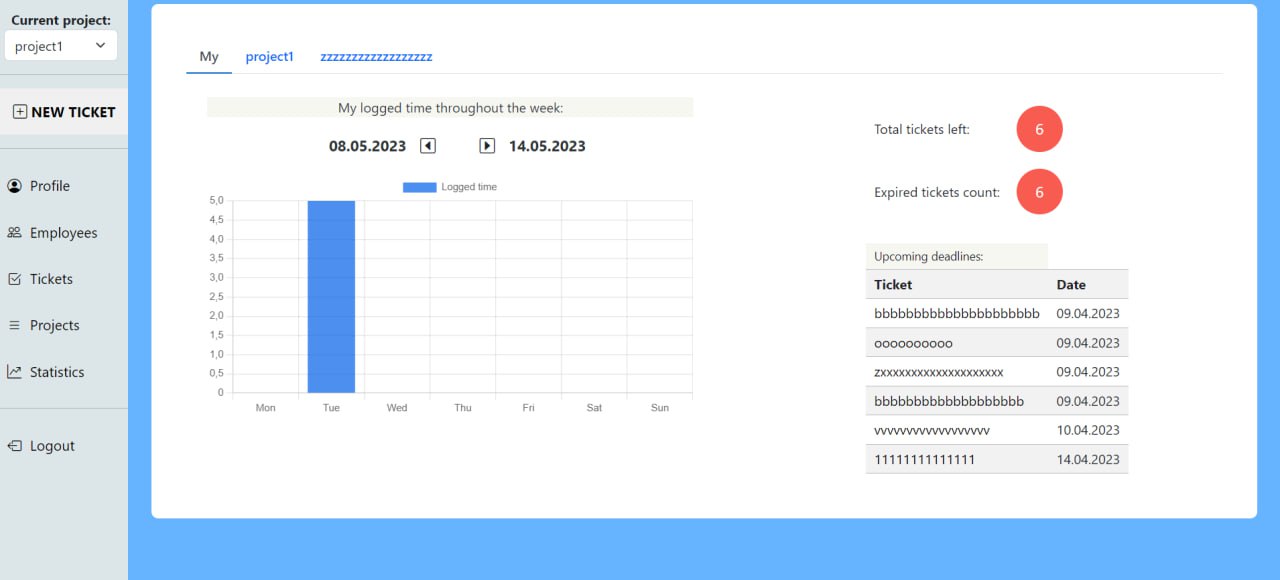


Рисунок 5.17 – Пример личной статистики сотрудника

На рисунке 5.18 представлен пример страницы со статистикой для отдельного проекта. Слева находится круговая диаграмма, отображающая количество задач по каждому статусу на данный момент. При наведении на каждую из секций будет показано точное число. Справа находится график, отображающий по дням количество завершенных задач в течение конкретной недели.



Рисунок 5.18 – Пример статистики по проекту

Крайней кнопкой в навигационном меню является выход из приложения. Она очистит все куки в браузере и направит пользователя на страницу для авторизации.

* 1. Руководство для менеджера

Менеджер – это сотрудник, но с расширенными правами. Он отвечает за работу с проектами и клиентами. Менеджер может выполнять все действия, что и сотрудник, а также специфичные действия, доступные только ему, которые рассмотрим далее.

Навигационное меню для менеджера представлено на рисунке 5.19. Как можно заметить, здесь добавились разделы Clients и Orders.

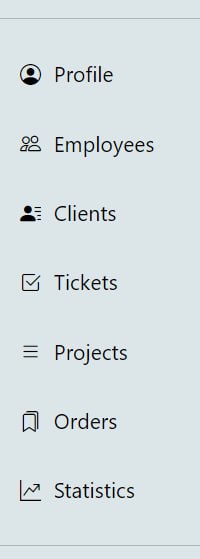


Рисунок 5.19 – Навигационное меню менеджера

При переходе на страницу Clients пользователь-менеджер может просмотреть список всех клиентов компании, а также осуществлять по ним поиск по имени и/или фамилии. Пример данной страницы можно будет просмотреть в руководстве для администратора на рисунке 5.23, но важно отметить, что для менеджера отсутствуют кнопки деактивации и блокировки пользователя.

Пример страницы Orders также представлен на рисунке 5.20. На данной странице представлены все заказы компании. Менеджер может блокировать заказы, однако после этого они не будут ему видны.



Рисунок 5.20 – Пример страницы с заказами

Если проект на основе какого-либо заказа уже существует, то можно нажать по соответствующей кнопке интерфейса и просмотреть его. Если проекта еще нет, то менеджер может создать его на основе выбранного заказа. Пример страницы для создания проекта представлен на рисунке 5.21.

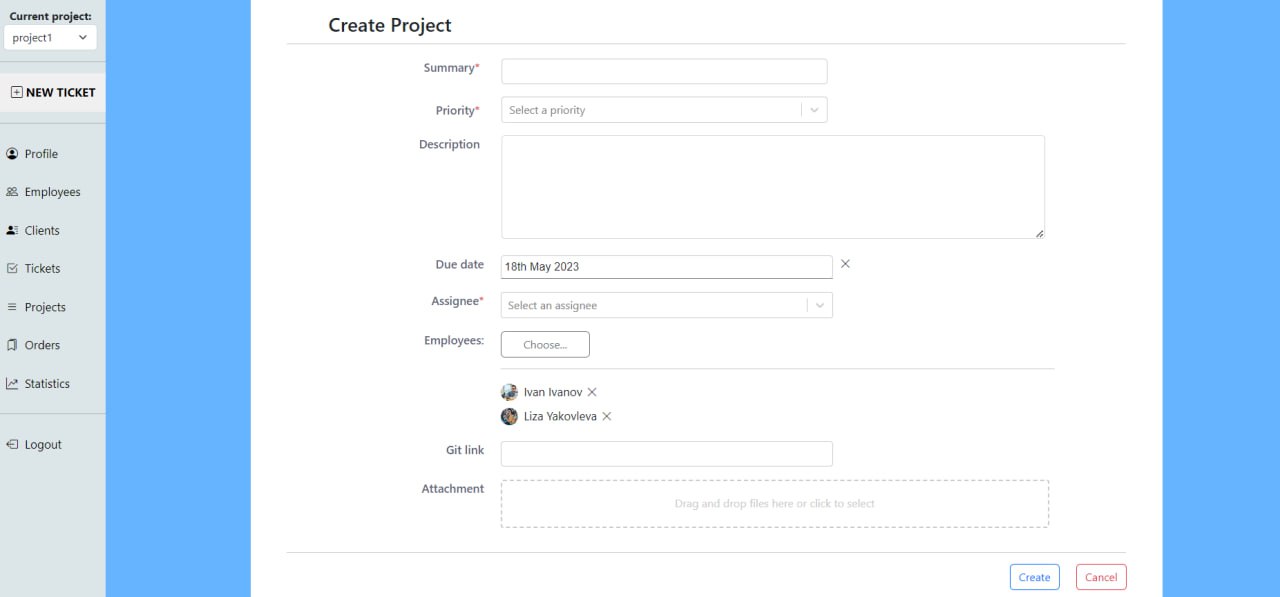


Рисунок 5.21 – Пример страницы для создания проекта

Для прикрепления сотрудников к проекту менеджер может выбрать их в специальном модальном окне, нажав на соответствующую кнопку.

Таким образом, только менеджер может создать проект. Редактировать проект может только тот менеджер, который его и создал, а также администратор.

* 1. Руководство для администратора

Администратор обладает неограниченными правами в приложении. Ему доступно все, что доступно любому сотруднику или менеджеру, а также некоторые расширенные возможности. Навигационное меню администратора идентично меню менеджера. Администратор – единственный, кто может создавать пользователей-сотрудников. Для этого необходимо перейти на страницу Users, где только для администратора будет отображена специальная кнопка для добавления пользователя. После нажатия на экране появится модальное окно, представленное на рисунке 5.22.

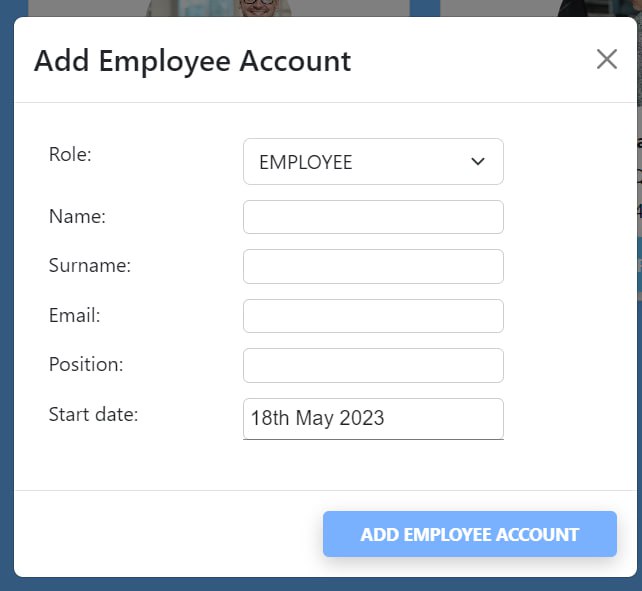


Рисунок 5.22 – Модальное окно для создания сотрудника

Пример отображения для администратора страницы со всеми клиентами представлен на далее рисунке 5.23.

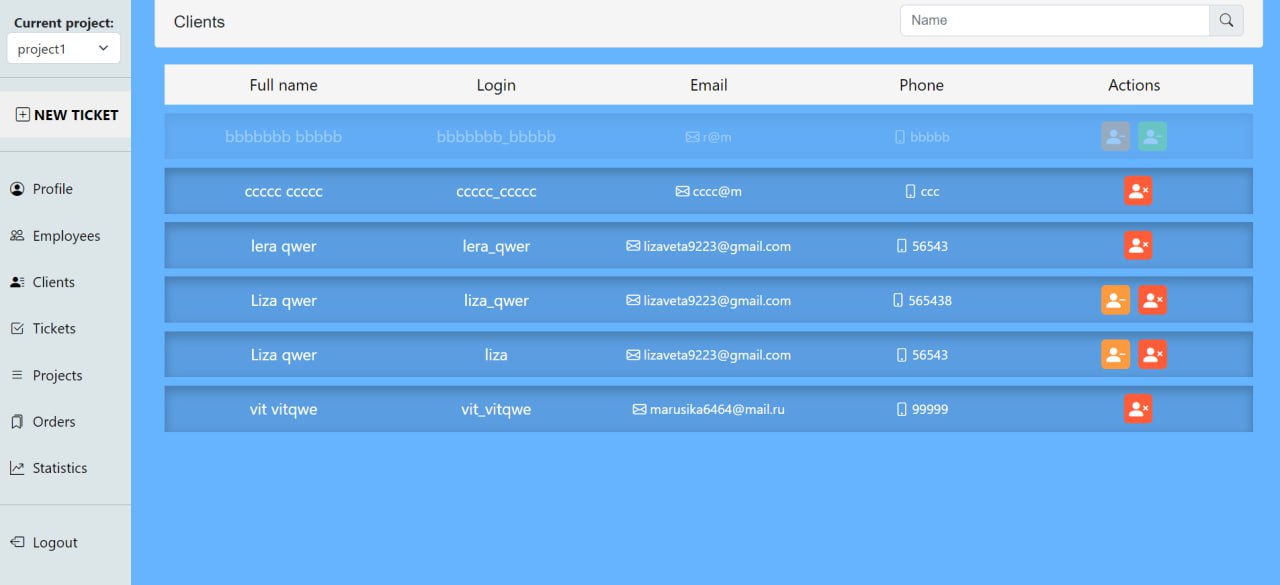


Рисунок 5.23 – Отображение для администратора страницы со всеми клиентами

Как можно увидеть, администратор обладает возможностью деактивации, блокировки и разблокировки клиентом. Аналогичное он может осуществлять и с пользователями-сотрудниками. Пример отображения страницы сотрудника для администратора представлен на рисунке 5.24. Как можно увидеть, администратору на ней доступны кнопки деактивации и блокировки.

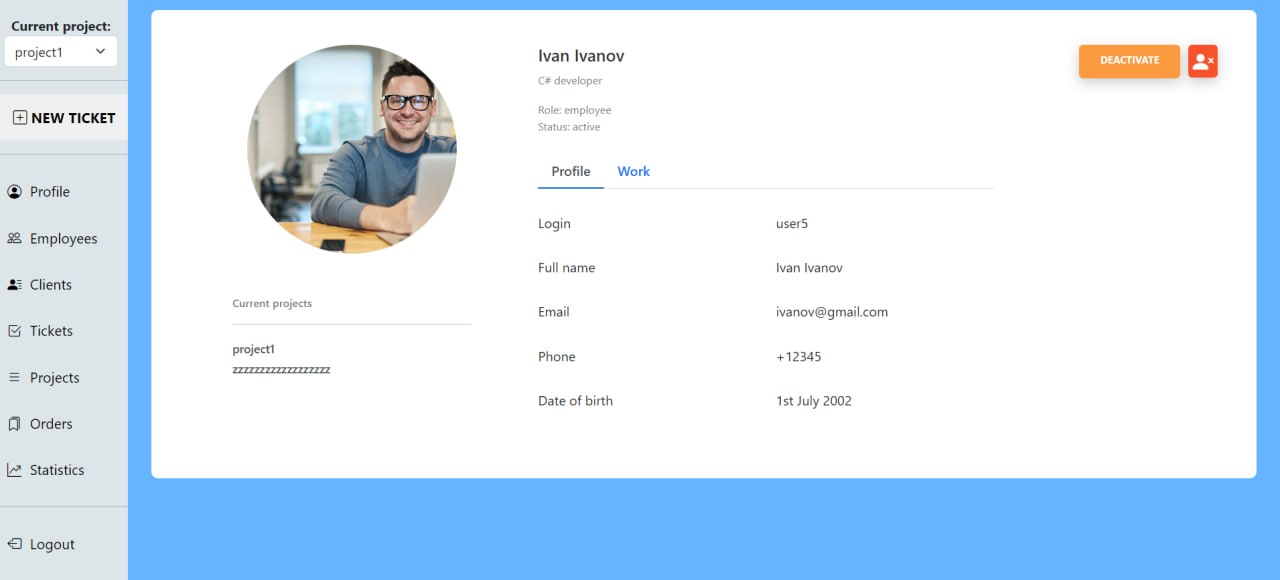


Рисунок 5.24 – Отображение страницы сотрудника для администратора

В случае деактивации пользователя, ему на электронную почту будет отправлено письмо с новым временным сгенерированным паролем, и затем при первом входе в приложение после деактивации, пользователю будет необходимо сменить временный пароль. Также администратор может видеть заблокированных сотрудников и клиентов на страницах со всеми сотрудниками/клиентами. Пример отображения представлен далее на рисунке 5.25.

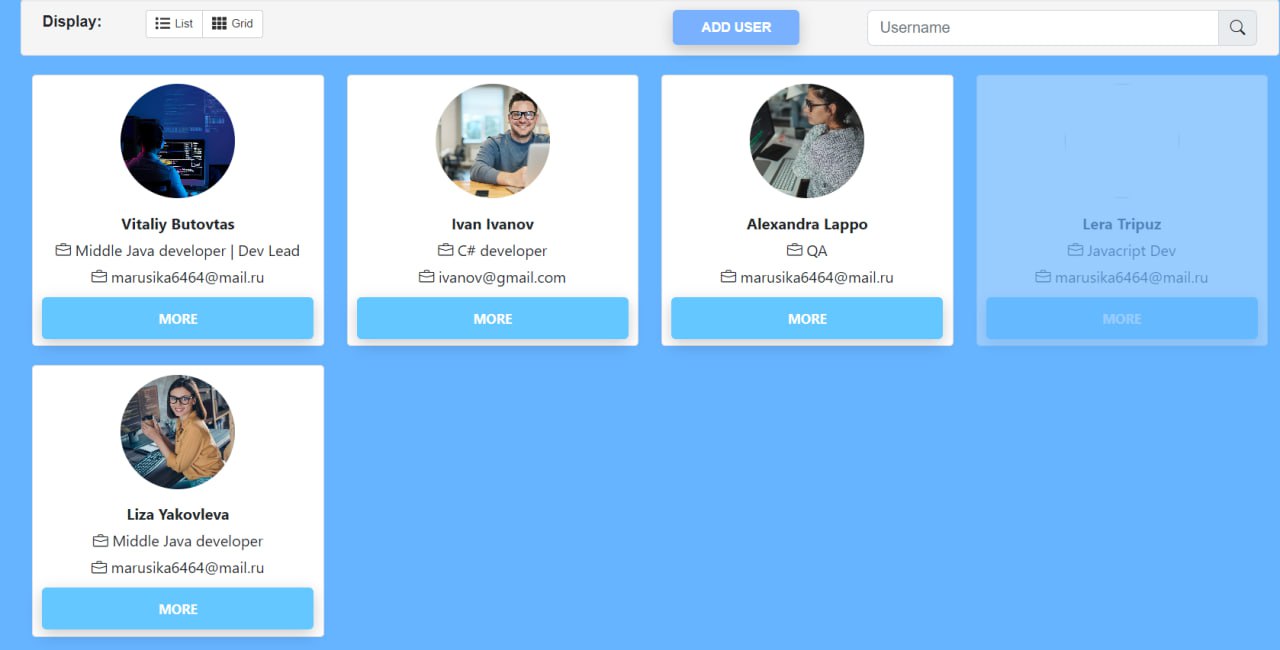


Рисунок 5.25 – Отображение заблокированных сотрудников

Как можно увидеть, заблокированные пользователи тут отображаются с приглушенным цветом. Также только администратор на странице Orders может видеть заблокированные заказы в приложении.

* 1. Руководство для клиента

Роль клиента создана для клиентов компании. Клиент может создавать заявки на выполнение проекта. Для этого ему необходимо перейти на общедоступный лендинг-сайт и заполнить там форму. Пример такой формы приведен на рисунке 5.26.

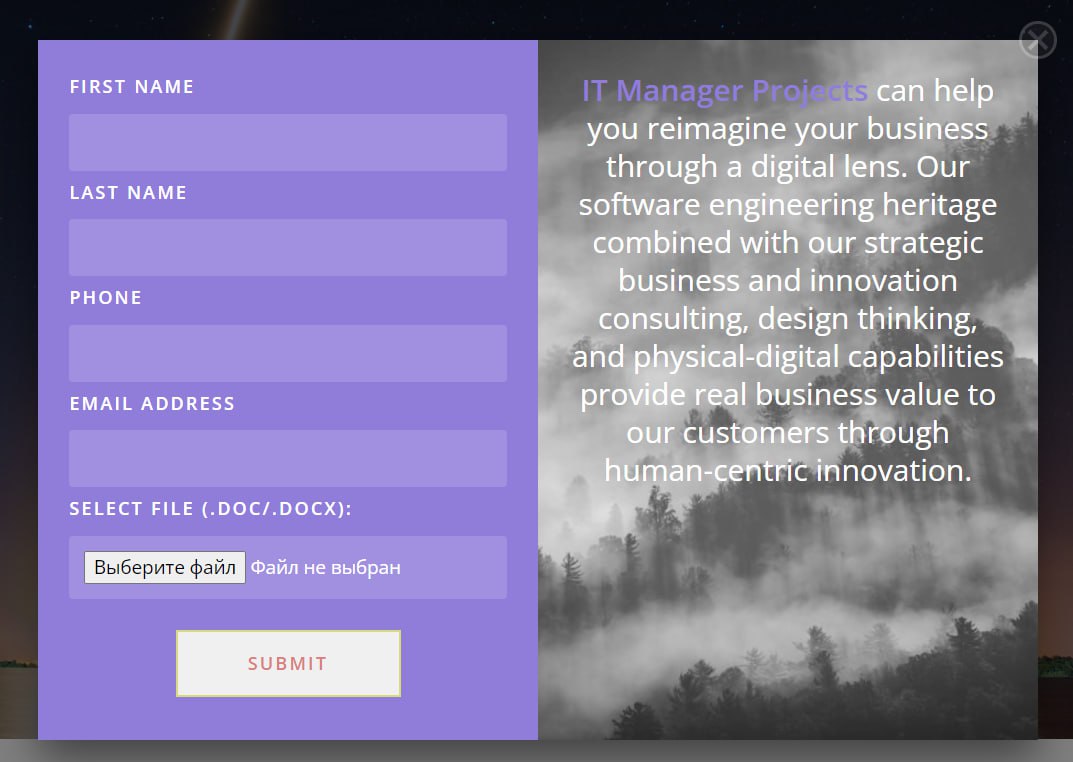


Рисунок 5.26 – Пример клиентской формы

После этого будет создана заявка клиента и автоматически будет сгенерирован профиль в приложении для него, а на почту будут высланы логин и пароль для входа. Однако пока у клиента не будет ни одного созданного проекта, у него не будет возможности войти в приложении. Профиль клиента представлен в виде модального окна, пример которого изображен на рисунке 5.27.

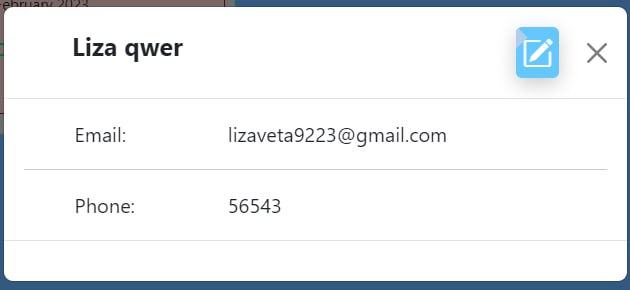


Рисунок 5.27 – Профиль пользователя-клиента

В приложении клиент может следить за выполнением своего проекта или проектов, если их несколько. Он может просматривать дэшборд по его проекту, детальную информацию по задаче этого проектам или по самому проекту. Клиент не может ничего редактировать или комментировать, а также просматривать чужие проекты или задачи, не относящиеся к его проекту. Клиент может просматривать статистику по своему проекту, однако у клиента соответственно нету личной статистики.

* 1. Выводы по разделу

Данный раздел предоставляет пояснения по работе пользователя с веб-приложением. Так как интерфейс приложения прост и интуитивно понятен, то ожидается, что у пользователя не должно возникнуть трудностей с его эксплуатацией.

Руководство содержит описание всех основных функций и возможностей приложения с приведением графических иллюстраций.

Данное приложение имеет 4 типа пользователя: сотрудник, менеджер, администратор и клиент. В данном разделе были описаны основные возможности каждой из ролей, цели создания этих ролей, а также методы их получения.

Помимо руководства по работе с приложением данный раздел включает в себя дополнительное пояснение по развертыванию приложения в среде Docker, а также инструкции по запуске на операционных системах Windows и MacOS.

Архитектуру приложения и большие функциональные возможности, предоставляемые пользователю, получилось ограничить небольшим количеством экранов, что является прямым доказательством хорошо спроектированного и простого пользовательского опыта в данном приложении. Но несмотря на это, руководство пользователя необходимо создавать всегда, независимо от сложности приложения, поскольку это не исключает возникновения недопонимания о пользователей.

1. Технико-экономическое обоснование проекта

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 06.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Семенова Л. С.

Утв.

Смелов В.В.

6 Технико-экономическое обоснование проекта

Лит.

Листов

8

*74218051, 2023*

У

Консульт.

Николайчук А.Н.

* 1. Общая характеристика разрабатываемого программного средства

При выполнении данного проекта было разработано программное средство, предназначенное для управления IT-проектами. Цель приложения – это обеспечить удобный менеджмент проектов и задач внутри IT-компании для ее сотрудников, а также клиентов. В приложении сотрудники в зависимости от своей должности могут создавать проекты, прикреплять к ним сотрудников, создавать и редактировать для проектов задачи, просматривать задачи в виде Kanban-доски и осуществлять поиск и фильтрацию по ним, комментировать задачи, логировать время, потраченное на выполнение задачи, осуществлять поиск среди пользователей, осуществлять поиск среди проектов компании, просматривать и редактировать свой профиль, просматривать личную статистику и статистику по конкретному проекту. Клиенты компании могут оставлять заявки на выполнение проектов и после начала разработки проектов следить за происходящим процессом.

Программное средство создано при помощи среды разработки IntelliJ IDEA и Visual Studio Code и языков программирования для серверной части Java и для клиентской части JavaScript. Для реализации функционала были использованы фреймворк Spring и библиотека React. В качестве базы данных использовалась СУБД MySQL. Для контейнеризации приложения был использован Docker. Для тестирования серверной части использовался Postman.

Разработанное программное решение имеет следующие преимущества перед рассмотренными в главе 1 аналогичными образами:

* возможность оставлять заявки на создание проектов для заказчиков;
* наличие профилей у пользователей, которые содержат полноценную расширенную информацию о сотруднике;
* осуществление поиска среди сотрудников;
* осуществление поиска среди проектов;
* логирование времени выполнения задач;
* возможность у заказчиков следить за ходом создания проекта;
* просмотр личной статистики и отдельно по проекту.

Приложение было создано на заказ и в последующем будет передано заказчику. Отпускная цена равняется среднерыночной стоимости аналогов.

Далее в этом разделе будет произведен маркетинговый анализ, расчет затрат на всех стадиях разработки программного средства и его определена себестоимость, то есть будет произведено полноценное технико-экономическое обоснование проекта.

* 1. Исходные данные для проведения расчетов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие нормативные правовые акты. Исходные данные для расчета приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчета

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Численность разработчиков, чел | Чр | 1 |
| Норматив дополнительной заработной платы, % | Ндз | 15 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34 |
| Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», % | Нбгс | 0,6 |
| Норматив накладных расходов, % | Нобп, обх | 55 |
| Ставка НДС, % | ННДС | 0 |
| Налог на прибыль, % | Нп | 0 |

Ставка НДС и налог на прибыль равны нулю, так как компания является резидентом ПВТ. Далее эти данные понадобятся для расчета всех необходимых параметров в процессе экономического анализа приложения.

Для оценки стоимости разработки необходимо проанализировать приложения, которые обладают аналогичным функционалом. Однако официальной информации о стоимости разработки рассмотренных в главе 1 аналогов компаниями не предоставлялось, и приблизительных данных также не найдено. Поэтому, чтобы рассчитать примерную стоимость аналогичных приложений, были использованы специализированные сайты для расчета ориентировочной стоимости программного обеспечения. Подробные данные приведены ниже в таблице 6.2.

Таблица 6.2 – Исходные данные аналогов

|  |  |  |
| --- | --- | --- |
| Источник | Стоимость, бел. руб. (без НДС) | Примечание |
| https://estimatemyapp.com/ | 15 000 | При оценке стоимостей были выбраны следующие параметры: это веб-приложение, средних габаритов, которое необходимо разработать с нуля, где пользователи могут авторизовываться, просматривать профили, просматривать дэшборд, осуществлять поиск, загружать файлы, оставлять комментарии, с наличием администраторских возможностей. |
| https://thebestapp.ru/calculator | 13 900 |
| https://digitalya.co/app-cost-calculator/estimate/ | 14 200 |

Таким образом, средняя цена разработки аналогичного продукта составляет 14366,67 белорусских рубля без учета НДС.

* 1. Обоснование цены программного средства

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как для разработчика, так и для пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижение таких показателей, как трудоемкость расчетов, алгоритмизация, программирование и отладка программ;
* сокращения расходов на оплату машинного времени, а также других ресурсов на отладку программных средств;
* снижения расходов на материалы;
* оптимизации программных средств;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* прочие прямые затраты;
* накладные расходы.

На основании данных затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

* + 1. Расчет затрат рабочего времени на разработку программного средства

В таблице 6.3 в укрупненном виде указаны все работы, выполненные для создания указанного в дипломной работе программного средства и количество рабочих часов, потраченных на выполнение этих работ.

Таблица 6.3 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, часов |
| 1 | 2 |
| Написание требований | 8 |
| Разработка дизайна интерфейса | 100 |
| Разработка архитектуры базы данных | 12 |
| Установка необходимых компонентов | 8 |

Продолжение таблицы 6.3

|  |  |
| --- | --- |
| 1 | 2 |
| Разработка архитектуры проекта | 8 |
| Разработка репозиториев серверной части | 20 |
| Разработка сервисов серверной части | 50 |
| Разработка контроллеров серверной части | 30 |
| Разработка страниц интерфейса | 80 |
| Разработка запросов между серверной и клиентской частями | 10 |
| Контейнеризация приложения | 10 |
| Программное тестирование | 20 |
| Ручное тестирование | 15 |
| Доработка после тестирования | 30 |
| Написание документации | 20 |
| **Всего** | **421** |

Итоговый результат по данной таблице будет использован далее для расчетов.

* + 1. Расчет основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработной платы full-stack разработчиков на Java. В итоге было установлено, что средняя месячная заработная плата на позиции junior+ составляет примерно 2100 рублей для Java full-stack разработчика, а часовая ставка составляет соответственно примерно 12,50 руб/час. Далее необходимо определить основную заработную плату исходя из трудозатрат и ЗП. Согласно таблице 6.3, проект разрабатывался одним специалистом на протяжении 421 час. Таким образом, основная заработная плата будет рассчитываться по формуле 6.1.

|  |  |  |
| --- | --- | --- |
|  | , | (6.1) |

где Соз – основная заработная плата, руб.;

Траз – время раработки (часов);

Сзп – средняя часовая ставка руб./час;

Краз – количество разработчиков, человек.

|  |  |  |
| --- | --- | --- |
|  | руб. |  |

В дальнейшем для других расчетов используется основная заработная плата, рассчитанная по указанной выше методике.

* + 1. Расчет дополнительной заработной платы

Дополнительная заработная плата представляет собой выплаты, предусмотренные законодательством о труде, включает компенсирующие выплаты (например, доплаты за работу в сверхурочное время, в государственные праздники, праздничные и выходные дни) и определяется по нормативу в процентах к основной заработной плате по формуле 6.2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | , | (6.2) | (6.4) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

Таким образом, дополнительная заработная плата составляет 789,38 руб.

* + 1. Расчет отчислений в Фонд социальной защиты населения и по страхованию

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний в БРУСП «Белгосстрах» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей. Отчисления в Фонд социальной защиты населения вычисляются по формуле 6.3.

|  |  |  |
| --- | --- | --- |
|  |  | (6.3) |

где – основная заработная плата, руб.;

– дополнительная заработная плата на конкретное ПС, руб.;

– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.4.

|  |  |  |
| --- | --- | --- |
|  |  | (6.4) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 36,31 руб., а в фонд социальной защиты населения – 2057,64 руб.

* + 1. Расчет суммы прочих прямых затрат

В таблице 6.4 приведены прочие затраты, которые были необходимы в ходе разработке дипломного проекта. Стоимость затрат указана за период разработки.

Таблица 6.4 – Прочие прямые затраты

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Статья затрат | | Стоимость, руб. |
| 1 | Среда разработки Intellij Idea Ultimate | | 526,53 |
| 2 | Доменное имя (.com) | | 14,5 |
| 3 | Веб-хостинг | | 19,2 |
| 4 | Редактор диаграмм Visio | | 14 |
|  | | **Всего** | **574,23** |

Таким образом, общая сумма прочих затрат составила 574,23 рубля.

* + 1. Расчет суммы накладных расходов

Сумма накладных расходов Собп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, рассчитывается по формуле 6.5.

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

Сумма накладных расходов составит:

= 5262,50 ⋅ 55 / 100 = 2894,38 руб*.*

Таким образом накладные расходы будут равны 2894,38 руб.

* + 1. Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы прочих затрат и суммы накладных расходов, по формуле 6.6.

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + Спз + Собп,обх. | (6.6) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = 5262,50 + 789,38 + + 36,31 + 574,23 + 2894,38 = 11614,43 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 11614,43 рублей.

Поскольку расходов на реализацию в данном случае нет, они будут соответственно равны нулю, и таким образом полная себестоимость проекта равна сумме расходов на разработку программного средства, то есть 11614,43 рублей.

* + 1. Определение цены, оценка эффективности

В данном разделе будет произведен расчет прибыли, которая будет получена после продажи приложения, а также будет оценена экономическая эффективность разработки данного проекта. Это итоговый и самый важный показатель, который отобразит целесообразность создания продукта.

Расчет будет отличаться в зависимости от целей разработки и способа монетизации. В данном случае продукт разрабатывается на заказ и передается заказчику.Эффект определяем по чистой прибыли, которая будет получена организацией-разработчиком при реализации продукта.

Расчета отпускной цены будет проводится по средней рыночной цене аналогов. Прибыль от реализации программного средства рассчитывается по формуле 6.7. Среднерыночная цена аналогов составляет примерно 14366,67 руб. без НДС. Прибыль от реализации программного средства рассчитывается как разница между отпускной ценой и полной себестоимостью продукта.

|  |  |
| --- | --- |
| , | (6.7) |
| ППС = 14366,67 – 11614,43 = 2752,24 руб. | |

Итого прибыль составила 2752,24 руб. при отпускной цене 14366,67 руб. без учета НДС и себестоимости равно 11614,43 руб.

Выше рассчитанное значение является величиной чистой прибыли, так как для резидентов ПВТ ставка налога на прибыль равна нулю. Для расчета уровня рентабельности продукта Урент воспользуемся формулой 6.8.

|  |  |  |
| --- | --- | --- |
|  |  | (6.8) |

|  |
| --- |
| У рент = 2752,24⋅ 100 / 11614,43 = 23,7 % |

Рентабельность рассчитывается как частное между прибыль от реализации программного средства и полной себестоимостью продукта, умноженное на сто процентов. Таким образом, как и значение чистой прибыли, уровень рентабельность является очень важным показателем и определяет экономическую эффективность проекта. В данном случае уровень рентабельности соствил 23,7% при прибыли в 2752,24 руб. и отпускной цене 11614,43 руб.

На данный момент, анализируя текущую ситуацию на IT-рынке, можно считать приемлемым уровень рентабельности примерно от 10 до 20%.

Исходя из этого, разработанное программное средство можно считать экономически эффективным, а его разработку соответственно целесообразной, учитывая все рассчитанные в данной главе экономические показатели.

* 1. Выводы по разделу

В таблице 6.5 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.5 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 421 |
| Количество разработчиков, чел. | 1 |
| Основная заработная плата, руб. | 5262,50 |
| Дополнительная заработная плата, руб. | 789,38 |
| Отчисления в Фонд социальной защиты населения и БРУСП «Белгосстрах», %, руб. | 2093,95 |
| Прочие прямые затраты, руб | 574,23 |
| Накладные расходы, руб | 2894,38 |
| Себестоимость разработки программного средства, руб. | 11614,43 |
| Полная себестоимость, руб. | 11614,43 |
| Чистая прибыль, руб | 2752,24 |
| Уровень рентабельности, % | 23,70 |

Поскольку величина рентабельности разработки программного средства, является положительным числом, то разработку программного продукта можно считать экономически целесообразной. Стоит отметить, что посчитать показатели прибыли и рентабельности для подобного продукта достаточно сложно, так как цены на подобные разработки не регламентируются и, наиболее оптимальным вариантом было использовать для расчетов среднюю цену на подобные разработки.

Разработка программного средства, осуществляемая одним программистом в течение двух с половиной месяцев при заданных условиях, обойдется компании в 11614,43 рублей. Реализация данного программного средства при среднерыночной цене в 14366,67 рублей принесет прибыль компании в размере суммы 2752,24 рубля. Уровень рентабельности составил 23,7%.

Благодаря этому приложению IT-компания может давать своим сотрудниками возможность эффективно и удобно заниматься управлением своих проектов.

Заключение

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Заключение

Лит.

Листов

1

*74218051, 2023*

У

Николайчук А.Н.

В ходе выполнения дипломного проекта были выполнены следующие изначально поставленные задачи для реализации функционала проекта:

* вся пользовательская рабочая информация cохраняется в базе данных;
* наличие возможности создания пользователей с заданными ролями;
* наличие возможности создания новых проектов;
* наличие возможности создания задач и их распределение по исполнителям;
* наличие возможности комментирования задач;
* наличие возможности просмотра личной статистики и статистики проекта.

Сначала были проанализированы схожие аналоги веб-приложения для управления IT-проектами, выявлены их положительные и отрицательные стороны. Результаты данного анализа были учтены при создании проекта.

Была спроектирована и разработана реляционная база данных для приложения, использующая экземпляр СУБД MySQL, имеющая в своем наборе 8 таблиц.

Для использования функционала созданной базы данных, был создан веб-сервис, написанный на языке программирования Java с использованием Spring Framework. Данный сервис содержит в себе все необходимые контроллеры позволяющими обслуживать пользовательские запросы, а также обеспечивает безопасность приложения благодаря авторизации пользователей на основе JWT-токенов.

Клиентская часть была разработана с использованием популярной библиотеки React, позволяющей создавать гибкую и удобную HTML-верстку, использовать различные библиотеки, а также легко взаимодействовать с AJAX-запросами.

Важным плюсом данного проекта является простота распространения, поскольку была добавлена возможность его распространения и запуска с использованием среды виртуализации Docker и Docker Registry.

Было проведено тестирование приложения на корректность работы по положительным и отрицательным сценариям, в ходе чего неисправностей выявлено не было.

Для конечных пользователей было написано руководство использования с подробным и наглядным описанием всех возможностей и функций.

Итоговым шагом было проведение экономического обоснования проекта, подтвердившее экономическую выгоду его разработки.

Таким образом, было создано веб-приложение, которое будет очень удобно для использования при менеджменте IT-проектов, а также легко для внедрения.

Список использованных источников

Ф.И.О

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Яковлева Е.И.*

Пров.

Кантарович В.С.

Н. контр.

Утв.

Смелов В.В.

Список использованной литературы

Лит.

Листов

2

*74218051, 2023*

У

Николайчук А.Н.

1. Trello [Электронный ресурс] / Atlassian – Режим доступа: https://trello.com/ – Дата доступа: 07.04.2023.
2. Notion [Электронный ресурс] / Notion – Режим доступа: https://www.notion.so/ – Дата доступа: 07.04.2023.
3. Jira [Электронный ресурс] / Atlassian – Режим доступа: https://www.atlassian.com/software/jira – Дата доступа: 07.04.2023.
4. Dynamic digital calendar day Kanban board: пат. 10496962 B1 США / Bryan Gregory Bartels, Patrick Humpal; заявитель Bryan Gregory Bartels, Patrick Humpal –   
   № : 15/995,156; заявл. 01.06.18; опубл. 03.12.19.
5. Kanban-based work processing device and work processing method: пат. 2022/0327486 A1 США / Jae Ho Choi; заявитель Jae Ho Choi, DELTA PDS CO. LTD – № : 17 / 333,734; заявл. 28.05.21; опубл. 13.10.22.
6. Встраиваемые системы и интернет вещей: почему надо учить Java в 2023 году [Электронный ресурс] / Synergy Times: медиа об образовании и карьере – Режим доступа: https://synergytimes.ru/evolve/vstraivaemye-sistemy-i-internet-veshchey-poche  
   mu-nado-uchit-java-v-2023-godu – Дата доступа: 08.04.2023.
7. Фреймворки в веб-разработке [Электронный ресурс] / Web creator: сложные IT-проекты и автоматизация бизнеса – Режим доступа: https://web-creator.ru/articles/about\_frameworks – Дата доступа: 10.04.2023.
8. Spring Boot 101: Введение в создание веб-приложений [Электронный ресурс] / VC.RU – Режим доступа: https://vc.ru/u/1389654-machine-learning/586955-spring-boot-101-vvedenie-v-sozdanie-veb-prilozheniy – Дата доступа: 10.04.2023.
9. Что такое реляционная база данных? [Электронный ресурс] / AWS – Режим доступа: https://aws.amazon.com/ru/relational-database/ – Дата доступа: 10.04.2023.
10. Идентификация, аутентификация и авторизация [Электронный ресурс] / Kaspersky Daily – Режим доступа: https://www.kaspersky.ru/blog/identification-authentication-authorization-difference/29123/ – Дата доступа: 10.04.2023.
11. Что такое контейнеризация? [Электронный ресурс] / Корпоративный облачный провайдер – Режим доступа: https://www.cloud4y.ru/blog/overview-of-containerization-technology/ – Дата доступа: 10.04.2023.
12. Развертывание нескольких контейнеров с помощью Docker Compose [Электронный ресурс] / Microsoft Build – Режим доступа: https://learn.microsoft.com/ru-ru/azure/cognitive-services/containers/docker-compose-recipe – Дата доступа: 12.04.2023.
13. Java Downloads [Электронный ресурс] / Oracle – Режим доступа: https://www.oracle.com/java/technologies/downloads/#jdk20-windows – Дата доступа: 17.04.2023.
14. Download IntelliJ IDEA [Электронный ресурс] / IntelliJ IDEA – Режим доступа: https://www.jetbrains.com/idea/download/#section=mac – Дата доступа: 17.04.2023.
15. Maven Repository [Электронный ресурс] – Режим доступа: https://mvnrepository.com/ – Дата доступа: 17.04.2023.
16. Что такое RESTful API? [Электронный ресурс] / AWS – Режим доступа: https://aws.amazon.com/ru/what-is/restful-api/ – Дата доступа: 19.04.2023.
17. 401 Unauthorized [Электронный ресурс] / Developer Mozilla – Режим доступа: https://developer.mozilla.org/ru/docs/Web/HTTP/Status/401 – Дата доступа: 26.04.2023.

ПРИЛОЖЕНИЕ А

Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Б

Листинг классов

services:

backend:

build: ../backend/

ports:

- 8080:8080

container\_name: backend

healthcheck:

test: ["CMD", "curl", "-f", "http://backend:8080/"]

interval: 20s

timeout: 10s

retries: 10

start\_period: 60s

depends\_on:

mysql:

condition: service\_healthy

frontend:

build: ../bigproject\_UI/

ports:

- 3000:3000

container\_name: frontend

depends\_on:

backend:

condition: service\_healthy

mysql:

image: mysql:8.0.28

container\_name: mysql

volumes:

- "../backend/db.sql:/docker-entrypoint-initdb.d/1.sql"

- "../backend/DML.sql:/docker-entrypoint-initdb.d/2.sql"

- dbdata:/var/lib/mysql

environment:

MYSQL\_ROOT\_PASSWORD: system

healthcheck:

test: ["CMD", 'mysqladmin', 'ping', '-h', 'localhost', '-u', 'root', '-p$MYSQL\_ROOT\_PASSWORD']

interval: 20s

timeout: 10s

retries: 10

start\_period: 30s

volumes:

dbdata:

package com.project.integration.web.controller;

import com.project.integration.serv.services.LoggedTimeService;

import com.project.integration.web.convertor.PojoConverter;

import com.project.integration.web.formmodel.TimeForm;

import lombok.RequiredArgsConstructor;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.time.LocalDate;

@RestController

@RequestMapping(value = "/time")

@RequiredArgsConstructor

public class LoggedTimeController {

private final LoggedTimeService loggedTimeService;

@GetMapping(value = "/ticket/{ticketId}")

public Float getTicketLoggedTime(@PathVariable("ticketId") Integer ticketId) {

return loggedTimeService.getTicketLoggedTime(ticketId);

}

@PostMapping(value = "/{employeeId}")

public ResponseEntity<Void> updateTicketTime(@PathVariable("employeeId") Integer employeeId, @RequestBody TimeForm timeForm) {

loggedTimeService.logTime(employeeId, PojoConverter.convertTimePojoToDto(timeForm));

return new ResponseEntity<>(HttpStatus.OK);

}

@GetMapping(value = "/employee/{employeeId}")

public ResponseEntity<Float[]> getWeekLoggedTime(@PathVariable("employeeId") Integer employeeId, @RequestParam(name = "weekFirstDate") String weekFirstDate, @RequestParam(name = "weekLastDate") String weekLastDate) {

return new ResponseEntity<>(loggedTimeService.getWeekLoggedTime(

employeeId,

LocalDate.parse(weekFirstDate),

LocalDate.parse(weekLastDate)

), HttpStatus.OK);

}

}

package com.project.integration.serv.services;

import com.project.integration.dao.entity.Employee;

import com.project.integration.dao.entity.LoggedTime;

import com.project.integration.dao.entity.Ticket;

import com.project.integration.dao.repos.EmployeeRepository;

import com.project.integration.dao.repos.LoggedTimeRepository;

import com.project.integration.serv.dto.TicketDto;

import com.project.integration.serv.dto.TimeDto;

import com.project.integration.serv.exception.ServiceException;

import com.project.integration.serv.mapper.LoggedTimeMapper;

import lombok.RequiredArgsConstructor;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.stereotype.Service;

import java.sql.Date;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Map;

import java.util.Objects;

import java.util.stream.Collectors;

@Service

@ComponentScan("com.project.integration.serv")

@RequiredArgsConstructor

public class LoggedTimeService {

private final LoggedTimeRepository loggedTimeRepository;

private final TicketService ticketService;

private final EmployeeRepository employeeRepository;

private final LoggedTimeMapper loggedTimeMapper;

public Float getTicketLoggedTime(Integer ticketId) {

return loggedTimeRepository.getTicketLoggedTime(ticketId);

}

public void logTime(Integer employeeId, TimeDto timeDto) {

Employee employee = employeeRepository.findById(employeeId).orElse(null);

if (Objects.isNull(employee)) {

throw new ServiceException("Employee [id=" + employeeId + "] does not exist");

}

TicketDto ticketDto = ticketService.getTicket(timeDto.getTicketId());

TicketDto updatedTicketDto = new TicketDto();

updatedTicketDto.setEstimatedTime(timeDto.getEstimatedTime());

ticketService.updateTicket(updatedTicketDto, ticketDto.getId(), false);

if (Objects.nonNull(timeDto.getLoggedTime())) {

LoggedTime loggedTime = new LoggedTime();

loggedTime.setTicket(new Ticket(timeDto.getTicketId()));

loggedTime.setEmployee(new Employee(employeeId));

loggedTime.setLoggedTime(timeDto.getLoggedTime());

loggedTime.setCreatedAt(LocalDateTime.now());

loggedTimeRepository.save(loggedTime);

}

}

public Float[] getWeekLoggedTime(Integer employeeId, LocalDate weekFirstDate, LocalDate weekLastDate) {

Map<LocalDate, Double> loggedTimeMap = loggedTimeRepository.getWeekLoggedTime(employeeId, weekFirstDate, weekLastDate)

.stream()

.collect(Collectors.toMap(s -> ((Date) s.get("date")).toLocalDate(), s -> (Double) s.get("time")));

Float[] loggedTime = new Float[7];

int idx = 0;

for (LocalDate i = weekFirstDate; weekLastDate.isAfter(i) || weekLastDate.equals(i); i = i.plusDays(1), idx++) {

Double value = loggedTimeMap.get(i);

if(Objects.nonNull(value)){

loggedTime[idx] = value.floatValue();

} else {

loggedTime[idx] = 0f;

}

}

return loggedTime;

}

}

package com.project.integration.dao.repos;

import com.project.integration.dao.entity.LoggedTime;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

import java.time.LocalDate;

import java.util.List;

import java.util.Map;

@Repository

@ComponentScan("com.project.integration")

public interface LoggedTimeRepository extends JpaRepository<LoggedTime, Integer> {

@Query(nativeQuery = true, value = "select sum(logged\_time.logged\_time) from logged\_time where logged\_time.ticket\_id = ?1")

Float getTicketLoggedTime(Integer ticketId);

@Query(nativeQuery = true, value = "select date(logged\_time.created\_at) as date, TRUNCATE(sum(logged\_time.logged\_time),1) as time " +

"from logged\_time where logged\_time.employee\_id = ?1 " +

"and logged\_time.created\_at >= ?2 and logged\_time.created\_at <= ?3 group by date(logged\_time.created\_at)")

List<Map<String, Object>> getWeekLoggedTime(Integer employeeId, LocalDate weekFirstDate, LocalDate weekLastDate);

}

const validateTicket = (event) => {

let isValidated = false;

if(selectedTtOption === null ||

selectedPriorityOption === null ||

selectedTtOption === null){

isValidated = true;

}

if (ticketName === '') {

isValidated = true;

setTicketNameErrMessage("Summary can't be empty");

} else if (ticketName.length > 100 ){

isValidated = true;

setTicketNameErrMessage("Summary max length is 100 symbols");

}

if((!gitLink && gitLink.length > 200) ||

!description && description.length > 1000){

isValidated = true;

}

if(isValidated){

event.stopPropagation();

setValidated(true);

}

return isValidated;

}

import React from "react";

import { Routes, Route, Navigate, useNavigate} from 'react-router-dom';

import { useCookies } from 'react-cookie';

import jwt\_decode from "jwt-decode";

import Authorization from './Authorization';

import Main from './Main';

import 'bootstrap/dist/css/bootstrap.min.css';

const isUserLoggedIn = (token) => {

if (token==null || token==undefined || token ==="undefined" || token === '') {

return false;

} else{

try{

let decodedToken = jwt\_decode(token);

if (decodedToken.exp > new Date().getTime()) {

return false;

}

else return true;

} catch(error){

return false;

}

}

};

const ProtectedRoute = ({ redirectPath = '/login', children }) => {

const [cookies] = useCookies(["token"]);

document.cookie = "url=" + window.location.pathname + "; path=/";

if(isUserLoggedIn(cookies.token)) return children;

else{

document.cookie = "token=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";

document.cookie = "project=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";

document.cookie = "employeeId=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";

return <Navigate to={redirectPath} replace />;

}

};

const GoToLogin = ({ children }) => {

const [cookies] = useCookies(["token", "url"]);

if(isUserLoggedIn(cookies.token))

return <Navigate to={cookies.url} replace />;

else return children;

};

function App(){

const navigate = useNavigate();

const [cookies] = useCookies(["token"]);

return(

<Routes>

<Route path="/\*" element={<Navigate to={isUserLoggedIn(cookies.token) ? "/app/\*" : "/login"} />} />

<Route path="/login"

element={

<GoToLogin>

<Authorization navigate={navigate} />

</GoToLogin>

}/>

<Route path="/app/\*"

element={

<ProtectedRoute>

<Main navigate={navigate} />

</ProtectedRoute>

}/>

</Routes>

);

}

const OnlyManagerAdminRoute = ({ redirectPath = '/', children }) => {

const [cookies] = useCookies(["token"]);

let decodedToken = jwt\_decode(cookies.token);

if(decodedToken.role==="ROLE\_MANAGER" || decodedToken.role==="ROLE\_ADMIN") return children;

else return <Navigate to={redirectPath} replace />;

};

export default function Main(){

return(

<div className="main">

<Routes>

<Route path="/\*" element={<Navigate to={decodedToken.role!==" CUSTOMER" ? "profile/" + cookies.employeeId : "dashboard"} />} />

<Route path="profile/:id" element={<Profile navigate={navigate} /> } />

{

<Route path="dashboard" element={<Dashboard navigate={navigate} listenCookieChange={listenCookieChange} /> } />

}

<Route exact path="ticket/:id" element={<SingleTask navigate={navigate} /> }

<Route path="projects" element={<Projects navigate={navigate} /> } />

<Route path="users" element={<Users navigate={navigate} /> } />

<Route path="project/:id" element={<ProjectPage navigate={navigate} /> } />

<Route path="statistics" element={<Statistics navigate={navigate} /> } />

<Route path="orders" element={

<OnlyManagerAdminRoute>

<Orders navigate={navigate} />

</OnlyManagerAdminRoute>

}/>

<Route path="orders/:id/create" element={

<OnlyManagerAdminRoute>

<CreateProject navigate={navigate} />

</OnlyManagerAdminRoute>

}/>

<Route path="clients" element={

<OnlyManagerAdminRoute>

<Clients navigate={navigate} />

</OnlyManagerAdminRoute>

}/>

</Routes>

</div>

);}

ПРИЛОЖЕНИЕ В

Логическая схема базы данных

ПРИЛОЖЕНИЕ Г

Блок-схема алгоритма перемещения задачи на «дэшборде»

ПРИЛОЖЕНИЕ Д

Блок-схема алгоритма изменения прикрепленного к задаче файла

ПРИЛОЖЕНИЕ Е

Блок-схема алгоритма авторизации заказчика в приложении

ПРИЛОЖЕНИЕ Ж

Блок-схема алгоритма изменения пароля от аккаунта

ПРИЛОЖЕНИЕ И

Структурная схема архитектуры приложения

ПРИЛОЖЕНИЕ К

Листинг классов SecurityConfig и JwtFilter

package com.project.integration.web.security;

import com.project.integration.serv.services.UserService;

import com.project.integration.web.exception.CustomAccessDeniedHandler;

import com.project.integration.web.exception.CustomAuthenticationEntryPoint;

import com.project.integration.web.security.jwt.JwtFilter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.access.AccessDeniedHandler;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private final JwtFilter jwtFilter;

private final UserService userDetailsService;

private final PasswordEncoder passwordEncoder;

private final CustomAuthenticationEntryPoint unauthorizedHandler;

@Autowired

public SecurityConfig(

JwtFilter jwtFilter,

UserService userDetailsService,

PasswordEncoder passwordEncoder,

CustomAuthenticationEntryPoint unauthorizedHandler) {

this.jwtFilter = jwtFilter;

this.userDetailsService = userDetailsService;

this.passwordEncoder = passwordEncoder;

this.unauthorizedHandler = unauthorizedHandler;

}

@Bean

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Bean

public AccessDeniedHandler accessDeniedHandler() {

return new CustomAccessDeniedHandler();

}

@Override

public void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.cors().and().httpBasic().disable().csrf().disable();

http.authorizeRequests()

.antMatchers("/", "/create\_request", "/auth", "/css/\*\*", "/js/\*\*")

.permitAll()

.antMatchers("/employee/new","/orders/{\\d+}/block", "/user/{\\d+}/block", "/user/{\\d+}/deactivate")

.hasRole("ADMIN")

.antMatchers( "/orders/all", "/orders/{\\d+}/description", "/project/create/{\\d+}")

.hasAnyRole("ADMIN", "MANAGER")

.antMatchers(HttpMethod.PUT, "/project/tickets/{\\d+}")

.hasAnyRole("ADMIN", "MANAGER", "EMPLOYEE")

.antMatchers(HttpMethod.POST, "/project/{\\d+}/tickets")

.hasAnyRole("ADMIN", "MANAGER", "EMPLOYEE")

.antMatchers("/project/reorder")

.hasAnyRole("ADMIN", "MANAGER", "EMPLOYEE")

.anyRequest()

.authenticated()

.and();

http.exceptionHandling()

.authenticationEntryPoint(unauthorizedHandler)

.accessDeniedHandler(accessDeniedHandler())

.and();

http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

}

}

package com.project.integration.web.security.jwt;

@Component

@RequiredArgsConstructor

public class JwtFilter extends OncePerRequestFilter {

private final JwtProvider jwtProvider;

@Override

protected boolean shouldNotFilter(HttpServletRequest request) {

return request.getRequestURI().equals("/")

|| request.getRequestURI().equals("/create\_request")

|| request.getRequestURI().equals("/auth")

|| request.getRequestURI().equals("/css/landing.css")

|| request.getRequestURI().equals("/js/landing.js"); }

@Override

protected void doFilterInternal( HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

throws ServletException, IOException {

String token = getTokenFromRequest(request);

String requestURL = request.getRequestURL().toString();

try {

if (StringUtils.hasText(token) && jwtProvider.validateToken(token)) {

if (requestURL.contains(URL.REFRESH))

throw new JwtException("Token is not expired yet");

UserDetails userDetails =

new UserDetailsImpl(

jwtProvider.getIdFromToken(token),

jwtProvider.getLoginFromToken(token),

Strings.EMPTY,

jwtProvider.getRolesFromToken(token),

jwtProvider.getStatusFromToken(token));

UsernamePasswordAuthenticationToken auth =

new UsernamePasswordAuthenticationToken(

userDetails, null, userDetails.getAuthorities());

SecurityContextHolder.getContext().setAuthentication(auth);

filterChain.doFilter(request, response);

return;

}

} catch (ExpiredJwtException e) {

if (requestURL.contains(URL.REFRESH)) {

if (jwtProvider.isRefreshAvailable(e)) allowForRefreshToken(e, request); else throw new JwtException("Token is not available for refresh yet"); } }

response.setStatus(HttpStatus.UNAUTHORIZED.value()); }

private String getTokenFromRequest(HttpServletRequest request) {

String bearerToken = request.getHeader(Attributes.AUTHORIZATION);

if (!StringUtils.hasText(bearerToken)) {

Cookie token = WebUtils.getCookie(request, Attributes.TOKEN);

if (Objects.nonNull(token)) bearerToken = "Bearer " + token.getValue(); }

if (StringUtils.hasText(bearerToken)

&& bearerToken.startsWith(Attributes.TOKEN\_BEGINNING\_IN\_HEADER)){

return bearerToken.replace(Attributes.TOKEN\_BEGINNING\_IN\_HEADER, Strings.EMPTY); } return Strings.EMPTY; } }