МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка системы оповещения пользователей сети

Обучающийся И.В. Орлов _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд.пед.наук, доцент, О.М. Гущина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Выпускная квалификационная работа посвящена разработке приложения для оповещения и управления задачами сотрудников.

Для системы управления задачами разработана клиентская и серверная части приложения. Клиентская часть включает в себя набор веб-страниц, отображаемых пользователю в браузере. Серверная часть осуществляет обработку запросов и взаимодействие с базой данных. Приложение позволяет добавлять, удалять, изменять и копировать задачи. А также позволять осуществлять просмотр списка задач сотрудника и задач его подчиненных, просмотр статистики по своим задачам и задачам сотрудников. Для реализации указанных действий на стороне клиентской части были разработаны веб-страницы.

Реализованная в рамках приложения система оповещения позволяет получать оповещения о новых задачах, об изменении даты и/или приоритета задачи, о переназначении задачи, об удалении, отмене и активации задачи. Содержимое оповещений соответствует данным о задаче.

Работа содержит введение, 3 главы, заключение и список использованной источников и состоит из 77 страниц. Так же в ней имеются 10 таблиц и 31 рисунок.

Оглавление

Введение

На сегодняшний день системы оповещения являются неотъемлемой составляющей нашей жизни и используются во многих сферах деятельности человека, не исключая информационные технологии. Основным назначением систем оповещения в сфере информационных технологии является уведомление пользователей программного продукта. Программным продуктом может быть, например, приложение для персонального компьютера, мобильное приложение или веб-приложение, работающее в браузере.

Способы и средства для реализации системы оповещения для каждого типа приложения свои. Нами рассмотрены способы оповещения пользователей веб-приложения. Наибольший интерес представляет реализация такой системы оповещения, которая позволит доставлять важные и срочные оповещения без потери их актуальности.

Рассмотрим степень изученности проблемы. Так как системы оповещения являются дополнительной составляющей программного продукта, то необходимо исследовать средства для реализации основного приложения. Основные вопросы инженерии корпоративного программного обеспечения и ее принципы рассмотрены в [1]. Реализация программного обеспечения с использованием языка программирования Java описаны в [2] [4][5][10][24]. Учитывая, что в реализации программного продукта могут появляться часто встречающиеся проблемы реализации, существуют уже готовые решения, называемые паттернами проектирования, которые рассмотрены в [3][23]. Вопросы, связанные с понятием системы оповещения, классификаций уведомлений, отображены в [6][16]. Поэтапный процесс разработки системы оповещения с использованием новых технологий приведен в [14]. Вопросы использования браузерных push-уведомлений, а также исследования технологий для их реализации рассмотрены [7][8][22][25].

Особое внимание уделяется технологиям организации и проектировании системы оповещения для срочных и важных сообщений. Однако в данных работах не рассматривались вопросы, связанные с оповещением пользователей веб-приложения, в частности с точки зрения реализации системы оповещения для веб-сайта некоторые организации уже предоставляют готовые решения, для которых требуется дополнение минимальной реализации от разработчиков веб-сайта. Также не рассматривались вопросы о проектировании и разработке системы оповещений, осуществляющих отправку оповещений, направленных на конкретного пользователя и динамическом формировании их содержимого. Все вышеизложенное обуславливает актуальность бакалаврской работы.

Совокупность системы оповещения и системы управления задачами сотрудников образуют автоматизированную систему для оповещения и управления задачами сотрудников.

Целью работы является реализация автоматизированной системы оповещения для управления задачами сотрудников.

Бакалаврская работа заключается в реализации системы для веб-приложения, осуществляющей доставку срочных и важных оповещений. Содержимое оповещений формируется динамически релевантно типу уведомления и функционирует независимо от того открыта ли страница веб-приложения.

В соответствии с целью были определены следующие задачи:

– исследовать способы оповещения пользователей в сети;

– определить требования к системе оповещения сотрудников;

– в соответствии с предъявленными требованиями выбрать оптимальный способ оповещения;

– проанализировать технологии, необходимые для реализации системы оповещения и системы управления задачами;

– спроектировать программную модель для разработки приложения оповещения и управления задачами сотрудников;

– разработать приложение для оповещения и управления задачами сотрудников в виде веб-приложения;

– осуществить тестирование разработанного приложения.

Выпускная квалификационная работа состоит из трех глав.

В первой главе рассмотрены понятия оповещения, системы оповещения, а также предъявлены классификации оповещений. Также затрагиваются вопросы, связанные со способами оповещения пользователей в сети, их преимуществами и недостатками и исследованием каждого из рассмотренных способов уведомления.

Вторая глава посвящена концептуальному и логическому проектированию системы оповещения и системы управления задачами сотрудников, а также анализу технологий, необходимых для реализации приложения для оповещения и управления задачами сотрудников.

Модель разработанного приложения, описание структур таблиц базы данных, классов, основной алгоритм работы и графический интерфейс приложения приведены в третьей главе.

Глава 1 Исследование процесса оповещения пользователей сети

1.1 Анализ области применения и функций систем оповещения

На протяжении всей истории человечества применялись различные способы оповещения. Используя крик, жесты, свет и дым сигнальных костров, звуки труб и барабанов, письменные послания люди оповещали друг друга о неординарном событии. Делалось это для того, чтобы была возможность предпринять соответствующие действия. Чаще всего под оповещением понималось предупреждение о возможной надвигающейся опасности. Но не всегда сообщение могло дойти до адресатов, или в случае письма уведомление могло идти очень долго и теряло свою актуальность. С достижениями научно-технической революции пришли новые средства и способы для оповещения. Информация становилась более доступной и охватывала значительно большее количество людей. Сообщения несли информацию не только о надвигающейся опасности, но и о менее значимых событиях. С развитием современных технологий, таких как Интернет, оповещения, стали частью нашей повседневной жизни. На современном этапе развития технологий уведомления являются неотъемлемой частью системы оповещения. Системы оповещения представляет собой набор протоколов и процедур, которые могут включать как присутствие человеческого фактора, так и наличие компьютерных компонентов. Применение компьютерных технологий помогает автоматизировать процесс передачи информации, сделать доставку более надежной и быстрой.

В информационных технологиях, оповещение - это сообщение пользователю о том, что событие, представляющее для него интерес, произошло в приложении. Под системой оповещения понимается совокупность программных и аппаратных средств, обеспечивающая отправку и доставку оповещений. Такие системы внедряются в различные программные продукты для дополнительной поддержки контакта с пользователями [12]. То

есть система оповещения существует как часть некоторого приложения или системы. Под пользователем понимается лицо, участвующее в функционировании системы или использующее результаты ее функционирования. Рассылаемые уведомления можно классифицировать по разным характеристикам. Рассмотрим классификации по следующим признакам:

– цель,
– создатель.

По цели можно выделить следующие оповещения:

– предупреждение,
– информационное оповещение,
– сообщение об ошибке.

Под предупреждением в приложении понимается оповещение, которое сообщает о намерении выполнить какое-либо действие. Информационное оповещение представляет собой оповещение, содержащее любую информацию актуальную для пользователя. К информационным сообщениям можно отнести напоминания, новости, реклама.

По классификации – создатель, можно выделить следующие типы оповещений [6]:

– оповещения, созданные пользователем;
– контекстно-генерируемые оповещения;
– оповещения, созданные системой.

Описание каждого типа отображено в таблице 1.

В зависимости от целей использования системы оповещения она может реализовывать различные типы оповещений. Во многих приложениях присутствует система оповещения, в виде предупреждений и сообщений об ошибках. Но существуют приложения, для которых необходимо организовать еще и напоминания. Например, таким приложением может быть планировщик задач или календарь.

Таблица 1 - Типы и описание оповещений

Тип	Описание
Оповещения, созданные пользователем.	Такие уведомления создаются пользователем с использованием приложения для других пользователей. Они вызывают интерес особенно тогда, когда содержание направлено на конкретного пользователя.
Контекстно-генерируемые оповещения.	Оповещения генерируются приложением на основе разрешения пользователей. При разрешении пользователя приложение может получить, например, местоположение пользователя и, учитывая данную информацию, генерировать оповещение.
Оповещения, созданные системой.	Оповещения генерируются приложением при необходимости.

Рассмотрим следующие функции системы оповещения:

- Эскалация. Степень критичности события может измениться еще до его завершения. Иногда небольшая ошибка может повлечь за собой целый ряд негативных последствий. Эти последствия могут привести к событиям, которые требуют незамедлительной реакции. Например, не удается отправить электронное письмо по адресу, такую ошибку можно посчитать несущественной, и классифицировать ее как предупреждение. Тем не менее, с помощью отправки электронной почты можно осуществить проверку соединения с почтовым сервером, отвечающего за передачу сообщений электронной почты. Если проверку осуществить не удается, то такое событие может привести к более серьезным проблемам, которые требуют тщательного рассмотрения, и их решение может занять длительное время. Таким образом, оповещение об инциденте может послужить сигналом для детального рассмотрения ошибок на ранней стадии развития проблемы.

- Взаимодействие в режиме реального времени. В случае возникновения любой чрезвычайной ситуации, которая требует вмешательства человека, система оповещения может обеспечить способ приема обратной связи. Обратная связь может быть использована для определения ряда задач, которые система должна выполнять. Например, при получении уведомления о пожаре

пожарный сообщает, о том, что он получил сообщение, тем самым устраняя необходимость в дальнейших уведомлениях.

- Приоритеты уведомлений. Некоторые уведомления могут иметь большее значение, чем другие (например, оповещение о пожаре будет гораздо важнее, чем уведомление о новом сообщении в социальной сети). Такого рода уведомления должны перекрывать любые существующие уведомления с более низким приоритетом. Система должна учитывать приоритеты уведомлений и отображать их в соответствующем порядке.

- Списки получателей. Если система не может связаться с необходимым получателем уведомления, система должна найти другого человека для уведомления, как правило, путем сканирования предварительно определенного списка.

- Планирование. Пользователь может установить промежуток времени, в котором стоит его уведомлять, и система оповещения должна это учитывать. Система может принимать во внимания отдых, выходные, праздники, и так далее, чтобы оповещение поступало в то время, когда на него можно среагировать.

Во многих программных продуктах можно обнаружить систему оповещения. Так как уведомления способствуют взаимодействию с пользователями, стимулируют использование и повторные посещения приложения, поэтому наличие системы оповещения может стать важным рычагом роста популярности приложения. Но не все уведомления для пользователя являются важными и полезными, некоторые из них могут даже и раздражать пользователя. Поэтому важно не только иметь систему оповещения, которая предоставляет услуги для пользователей, также необходимо предоставить пользователю возможность решить то, о чем он действительно хочет получать уведомления. Поэтому при разработке важно понимать степени необходимости оповещений. Так как слишком частые и малоинформативные оповещения могут доставлять пользователю только неудобства [12].

Реализацию такой системы можно найти и в мобильных приложениях, и в приложениях, установленных на компьютере. Не исключением стали и веб-приложения. В последнее время система оповещения стала важным аспектом разработки веб-приложений. Использование такой системы в веб-приложении позволяет уведомлять пользователей в сети. Веб-приложением называется клиент-серверное приложение, для которого клиентом является браузер, а сервером - веб-сервер. Браузером называется программное обеспечение для просмотра веб-страниц, а также управления веб-приложениями и решения других задач. Для реализации такого приложения требуется разработки клиентской и серверной частей. Клиентская часть веб-приложения представляет собой графический интерфейс, отображаемый в браузере. На стороне клиента формируются запросы на сервер и обрабатываются ответы на эти запросы. Серверная часть веб-приложения - это программа, реализующая бизнес-логику приложения, осуществляющая обработку запросов клиента и формирование ответов на запросы, а также работу с базой данных. Клиентская и серверная часть общаются по сети посредство протокола HTTP.

Разработка веб-приложения с системой оповещения является одной из актуальных проблем. Ведь существующие технологии для реализации такой системы не всегда являются оптимальными. Для определения наиболее подходящего способа оповещения необходимо рассмотреть все возможные варианты и определить, какие из них подходит для веб-приложения. Выбор способа оповещения зависит от того, какие оповещения необходимо будет рассылать, будь то срочные сообщения или содержащие информацию рекламного характера.

Далее мы рассмотрим основные способы уведомления, какие средства при этом используются, а также их достоинства и недостатки.

1.2 Исследование основных способов оповещения в информационных технологиях

К основным способам уведомления можно отнести телефонные звонки, рассылка электронной почты, СМС-рассылка, всплывающие окна и push-уведомления. Рассмотрим каждый из перечисленных способов более подробно.

Телефонный звонок является ярчайшим представителем голосовых оповещений из перечисленных вариантов и наиболее очевидным. Можно привести следующие примеры. Сотовые операторы уведомляют своих клиентов о появлении новой услуги или тарифа. Очевидно, что если будет необходимость оповещать достаточно часто, то данный способ не является оптимальным.

Следующий способ - оповещение с помощью рассылки по электронной почте. Электронной почтой называется система обмена электронными сообщениями, называемыми электронными письмами, между пользователями компьютерной сети, в том числе и Интернета. Рассылка электронной почты представляет собой средство массового, группового общения, рекламы и оповещения. Она заключается в автоматизированной отправке сообщений электронной почты группе адресатов из заранее составленного списка.

Рассылка является инструментом маркетинга и рекламы. При помощи рассылки электронной почты можно стимулировать продажи, поддерживать интерес клиентов к компании или товарам. Но существование такого явления, как спам, является проблемой для информирования с помощью электронной почты. Во-первых, потому что это письма нежелательные, они могут доставить пользователю массу неудобств. Во-вторых, очень часто спам используется для передачи вредоносных файлов, что может повлечь за собой немало проблем. В-третьих, спам осуществляет постоянную нагрузку на почтовые сервера, и как следствие этого возникает угроза стабильности их работы.

Рассылки по электронной почте на сегодняшний день является популярным способом оповещения. Но данный способ не всегда оптимален, например, для важных и срочных сообщений и новостей он не является подходящим.

Ниже в таблице 2 отображены преимущества и недостатки рассылки электронной почты.

Таблица 2 - Типы формирования списка рассылки

Тип формирования	Определение
Список рассылки	Сервер получает сообщение от любого подписчика на специальный адрес, затем перенаправляет полученное сообщение всем подписчикам из списка рассылки.
Групповая рассылка	Позволяет иметь доступ нескольким пользователям к почте, поступающей на один адрес вне зависимости от отправителя.
Информационная, рекламная рассылка	Подготовленное сообщение отправляется одновременно всем подписчикам данной рассылки без возможности ответа на него. Если не предусмотрена предварительная подписка, такая рассылка называется спамом.

Помимо телефонных звонков, существует еще один способ оповещения с использованием мобильных телефонов – СМС-рассылка. СМС (служба коротких сообщений) технология, которая позволяет осуществлять прием и передачу коротких текстовых сообщений с помощью мобильного телефона. Так как для оповещения пользователя по СМС требуется информация о номере телефона, важно помнить об обеспечении безопасного хранения таких данных, чтобы злоумышленник не смог похитить номера телефонов пользователей. Для работы не требуется доступа к Интернету, поэтому такой способ оповещения подходит для важных, а также срочных сообщений. Можно привести несколько примеров использования таких оповещений. Сервисы, связанные с осуществлением финансовых операций, для подтверждения действия отправляют код на телефон. Полученный код

необходимо ввести в поле на сайте, тем самым давая согласие на проведение операции.

Еще одним способом уведомления являются всплывающие окна. Всплывающим окном называется окно, которое открывается на экране устройства в результате выполнения какого-либо действия. Под окном понимается оконный интерфейс. Помимо оповещений и предупреждений, всплывающие окна имеют широкий диапазон применений. Всплывающие окна могут отображать не только браузеры, но и другие программы. Массовое применение всплывающих окон в целях рекламы в сети Интернет привело к появлению в браузерах функции блокирования всплывающих окон. Практически каждый современный браузер включает такую функцию, а также дает возможность пользователю создавать список сайтов, для которых показ всплывающих окон разрешен или запрещен. Для работы системы оповещения, использующей в качестве уведомления всплывающие окна, для веб-приложения, необходима открытая страница веб-приложения, что не всегда обеспечивает достаточную автономность системы оповещения. Для получения уведомления пользователь должен держать открытой страницу веб-приложения, что не является удобным, особенно если таких приложения с системой оповещений несколько. Но иногда такая функциональность является достаточной. Так, например, осуществляются уведомления в социальных сетях или сообщения об ошибках.

1.3 Push-уведомления как способ оповещения пользователей в сети

На данный момент push-уведомления является одними из наиболее распространённых уведомлений в сети. Для отправки таких уведомлений используется push технология, распространяющая информацию от сервера к клиенту. Характерная черта такой технологии заключается в том, что сервером отправляются данные клиенту без его предварительного запроса. Полной противоположностью push технологии является pull технология, где запрос

инициирует клиент. Система push-уведомлений реализуется с использованием push технологии. Сами же push-уведомления представляют собой короткие сообщения, которые владельцы сайтов или мобильных приложений могут рассылать своим пользователям. Push-уведомления делятся на браузерные и мобильные. Мобильные push-уведомления поддерживаются операционными системами Android и iOS. Что же касается браузерных уведомлений, они появились позже мобильных уведомлений и изначально не все браузеры их поддерживали.

Push оповещения состоят из заголовка, текста, изображения, а в случае браузерных еще и ссылки на сайт. Мобильные push-уведомления появились раньше браузерных, и активно используются в мобильной разработке [17]. Для расширения круга использования push технологии, была проведена разработка и реализация браузерной версии. Компоненты, из которых состоит система push-уведомлений, не зависит от того, какие именно push-уведомления будут использоваться, браузерные или мобильные.

Для работы системы push-уведомлений используются следующие компоненты:

– сервер push-уведомлений;
– сервер приложений;
– служба, работающая в фоновом режиме;
– приложение, поддерживающее push-уведомления.

Рассмотрим функции перечисленных компонентов. FCM (Firebase Cloud Messaging) выполняет следующий перечень функций:

– осуществление взаимодействие с работающей в фоновом режиме службой;
– выдача идентификаторов устройствам или браузерам, на которых используется приложение;
– отправка уведомлений по запросу сервера приложений.

Для каждой мобильной операционной системы и браузера, которые поддерживают push-уведомления, существует сервис для рассылки

оповещений. Для операционной системы Android и браузера Chrome - это сервис FCM (Firebase Cloud Messaging), для мобильной операционной системы iOS и браузера Safari таким сервисом является APNS (Apple Push Notification Service). Перечисленные сервисы дают возможность отправлять уведомления от сторонних приложений, тем самым упрощая разработку системы оповещений для приложения. Так как не придется самостоятельно реализовывать функции сервера уведомлений [14].

Основной функцией сервера приложений является отправка запросов серверу уведомлений. Сервер приложений должен иметь SSL сертификат вне зависимости от того, какие push-уведомления будут реализованы, браузерные или мобильные. SSL расшифровывается, как Secure Socket Layer, представляет собой криптографический протокол, который используется для обеспечения безопасного соединения между сервером и приложением [15]. SSL соединение гарантирует защищенность передаваемой информации от злоумышленников. Для осуществления такого соединения необходимо, чтобы сервер имел установленный SSL сертификат. SSL сертификат является цифровым сертификатом, который в свою очередь представляет собой файл, идентифицирующий пользователей и серверы. Он проводит аутентификацию сервера перед установкой сеанса SSL соединения. Обычно цифровой сертификат подписывается и заверяется независимой третьей стороной, что является гарантией его достоверности [9].

Служба, работающая в фоновом режиме, принимает уведомления от сервера уведомлений и отображает их пользователю. Именно она позволяет получать уведомления даже, если приложение закрыто. Для мобильных приложений такой службой является сама операционная система. Наличие и корректная реализация всех перечисленных компонентов обеспечивает надежную работу системы. После рассмотрения компонентов и определения функций рассмотрим схему работы системы push-уведомлений. Она отображена на рисунке ниже (рисунок 1).

Рисунок 1 - Общая схема работы системы push-уведомлений

Предварительно сервер приложений регистрируется на сервере уведомлений. После соглашения пользователя на получение уведомлений приложение регистрируется у операционной системы, в случае мобильного приложения, или у браузера, в случае веб-приложения. Затем либо операционная система или браузер, отправляет запрос серверу уведомлений на получение идентификатора устройства. После получения ответа, устройству присваивается идентификатор. После этого приложение отправляет присвоенный идентификатор на сервер приложений. Сервер приложений сохраняет его, например, в базе данных. При наступлении момента для оповещения пользователя сервер приложений отправляет уведомление серверу уведомлений, а FCM отправляет службе, работающей в фоновом режиме. Мобильное приложение или Web приложение, в свою очередь, отображает оповещение пользователю.

Из рассмотренных способов оповещения, наиболее подходящим способом оповещения для веб-приложения являются браузерные push-

уведомления. Так как для их реализации не требуется дополнительной личной информации, такой как электронная почта и номер телефона. Push-уведомление не теряет свою актуальность, в отличие от оповещений по электронной почте. Такой способ является подходящим для всех типов оповещений, а также срочных сообщений и новостей. Для push-уведомления нет необходимости в открытой странице веб-приложения. Такой способ оповещение с использованием push-уведомлений является безопасным, так как дополнительно требуется наличие SSL сертификата у сервера приложений.

Предположим, что требуется разработать веб-приложение, представляющее собой систему управления задачами сотрудников предприятия. В такой системе начальник может создавать задачи подчиненному, а тот в свою очередь должен их исполнять. Для такой системы управления оповещения представляют особую значимость, так как в данном случае система оповещения позволит ускорить процесс взаимодействия начальника и подчиненного, тем самым сэкономив время. Начальник сможет получать уведомления о статусе исполняемой задачи, а подчиненный о назначении ему новой задач даже, если приложение закрыто. В этом случае очень важно, чтобы оповещения приходили быстро и не теряли свою актуальность.

Наиболее подходящим способом уведомления сотрудников будут браузерные push-уведомления, так как они будут доходить до сотрудника, даже если приложение закрыто.

Таким образом, можно определить следующие требования к разрабатываемой системе оповещений:

– система должна осуществлять быструю и надежную доставку уведомлений сотрудникам;

– система должна обеспечивать оповещение сотрудников о назначении, удалении, отмены или изменении задачи;

– система должна оповещать начальника при изменении подчиненным статуса задачи;

– при изменении исполнителя задачи система должна оповестить и нового исполнителя, и предыдущего.

Можно определить следующие функции, которые должна выполнять система управления задачами:

– аутентификация, авторизация и регистрация сотрудников;

– просмотр списка своих задач и задач подчиненных;

– добавление, удаление, изменение и копирование задач;

– просмотр задачи;

– просмотр статистики по задачам.

Выводы по главе 1

Мы пришли к выводам, что на сегодняшний день для уведомления о совершении каких-либо событий широко используются системы оповещения. Такие системы позволяют автоматизировать процесс информирования. Они используются в различных сферах деятельности человека. В том числе системы оповещений активно применяются в сфере информационных технологий, так как с ее помощью можно дополнить функциональность приложений и обеспечить взаимодействие с пользователями приложения. Системы оповещения можно реализовать для многих приложений, в том числе и для веб-приложений.

Актуальность использования браузерных push-уведомлений велика, за счет того, что для их реализации не требуется дополнительной личной информации, такой как электронная почта и номер телефона.

Push-уведомление не теряет свою актуальность, в отличие от оповещений по электронной почте. Такой способ является подходящим для всех типов уведомлений, а также срочных сообщений и новостей. Для получения push-уведомления нет необходимости в открытой странице веб-

приложения. Такой способ оповещений с использованием push-уведомлений является безопасным, так как дополнительно требуется наличие SSL сертификата у сервера приложений.

Нами выделены различные функции системы оповещения, реализация которых способствует совершенствованию процесса информирования. Выделены цели использования систем оповещения начиная от простого информационного сообщения и заканчивая предупреждением.

Для достижения каждой цели используют соответствующий способ оповещения.

Анализ функций системы оповещения и классификация таких систем позволили раскрыть актуальность внедрения системы оповещения как части веб-приложения.

Изучение способов оповещения позволило нам прийти к выводам о наиболее релевантных способах оповещений, применяемых в информационных технологиях.

Выделены требования, на основе которых в дальнейшем мы будем разрабатывать систему оповещений пользователей в сети.

Глава 2 Проектирование архитектуры системы оповещения

2.1 Концептуальный проект системы оповещения

В данной главе рассмотрим технологии и инструментарий, позволяющие реализовать систему управления задачами и систему оповещения сотрудников. После того как системы будут спроектированы и реализованы их совокупность составит веб-приложение по управлению задачами и оповещения сотрудников. Основная цель веб-приложения состоит в том, чтобы позволить пользователю выполнять следующие задачи:

– создание, добавление, удаление, копирование задач;

– мониторинг процесса выполнения задач;

– контроль показателей выполнения задач подчиненных;

– отправка и получение оповещений.

Определим элементы, которые лягут в основу будущего веб-приложения:

– инструментарий для реализации северной части, содержащей бизнес-логику приложения;

– среда разработки, которая будет использована для написания кода веб-приложения;

– инструмент для автоматизированной сборки приложения;

– способ хранения данных веб-приложения и организация их хранения;

– инструментарий для реализации клиентской части веб-приложения;

– способ взаимодействия между клиентом и сервером, а также сервером приложений и сервером уведомлений.

На этапе проектирования нами построены UML-диаграммы, такие как диаграмма вариантов использования (рисунки 2-3), диаграммы деятельности (рисунок 4), диаграммы последовательности (рисунок 5). В ходе создания

диаграммы вариантов использования выделено две роли: руководитель отдела и исполнитель задачи. На рисунке ниже (рисунок 2) представлена диаграмма использования системы оповещения и работы с задачами руководителем.

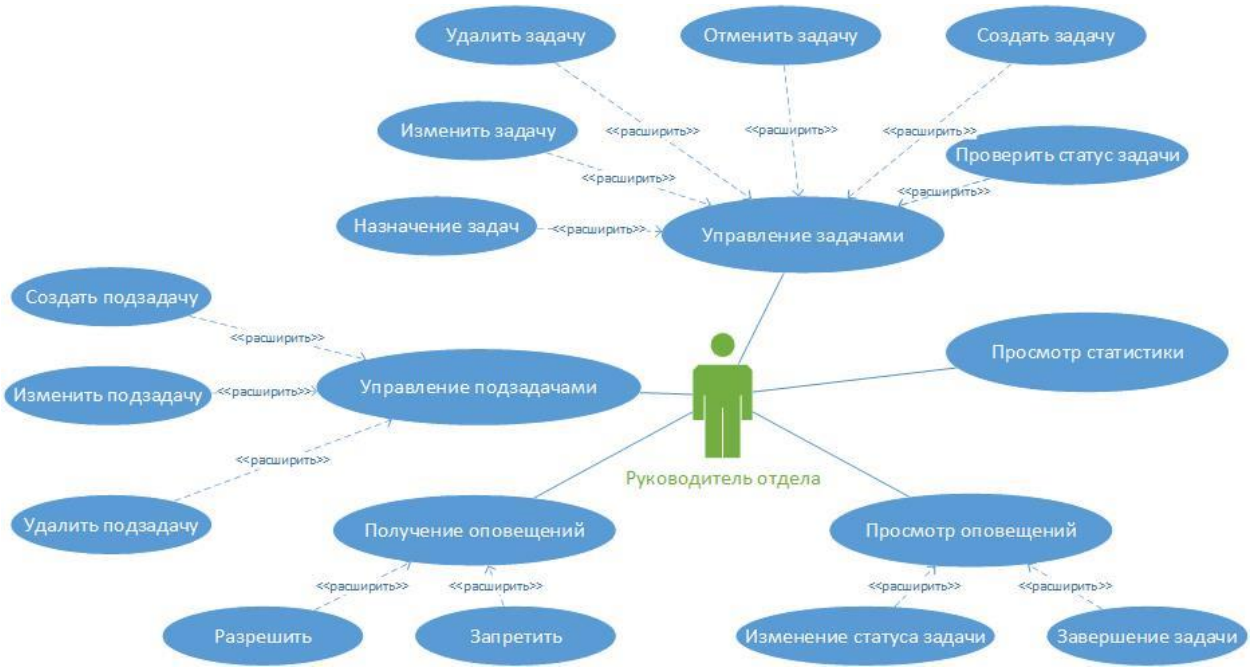

Рисунок 2 – Варианты использования системы руководителем

На рисунке 2 отображены следующие варианты использования системы в лице руководителя отдела:

– Управление задачами: руководителю отдела должен быть доступен функционал по созданию, назначению, изменению, удалению, отмене задач;

– Просмотр статистики. У руководителя должна быть возможность просмотра статистики по количеству назначенных и выполненных задач исполнителем;

– Просмотр оповещений. Руководитель может просмотреть полученное оповещение и проследить за изменением статуса задачи. Статус задачи может быть одним из перечисленных: «Назначена», «Взята в работу», «Выполнена», «Закрыта», «Отменена», «Возврат в работу»;

– Получение оповещений. Вариант использования «Получение оповещений» предполагает возможность разрешить и запретить получение оповещений от системы;

– Управление подзадачами. У руководителя должна быть возможности создать подзадачу в рамках задачи, что позволит детально проработать проблему.

Активным участником разрабатываемой системы оповещения пользователей является – исполнитель. Диаграмма вариантов использования системы исполнителем представлена на рисунке 3.

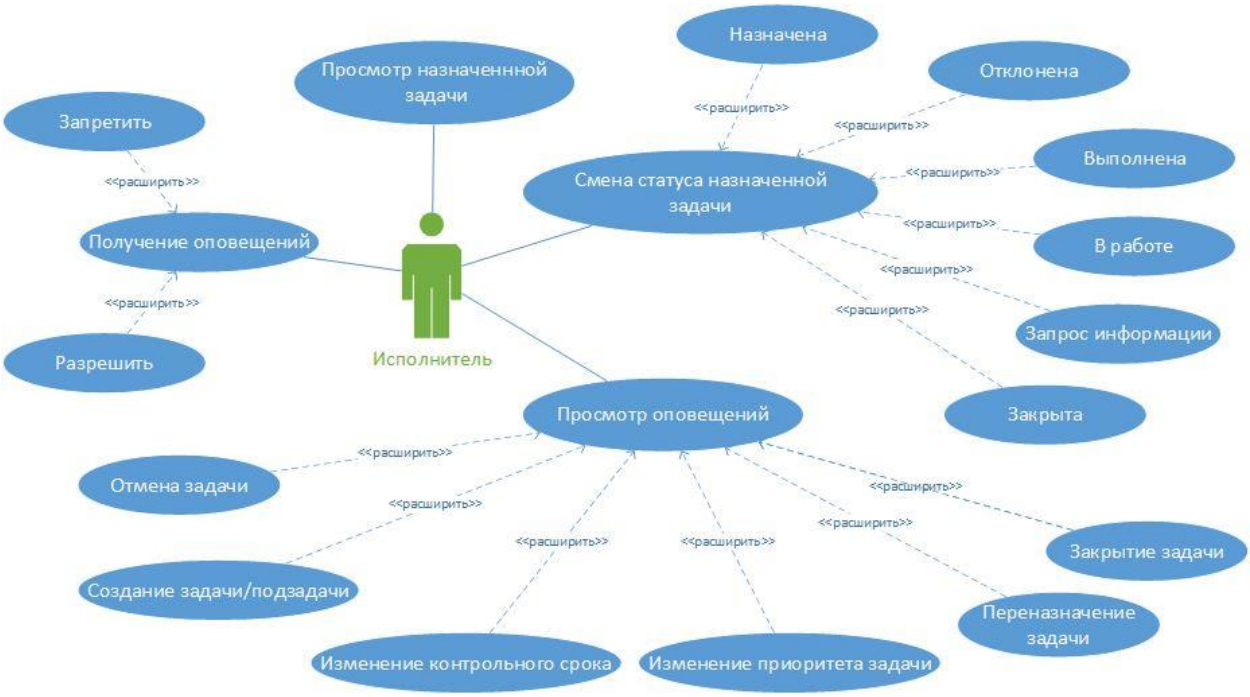

Рисунок 3 – Варианты использования системы исполнителем

Рассмотрим подробнее диаграмму вариантов использования исполнителем:

Просмотр назначенной задачи. Предполагает, что пользователь, войдя в систему, может посмотреть все назначенные на него задачи;

Смена статуса назначенной задачи. В процессе решения поставленной задачи предполагается изменение её статуса. При проведении первичного

анализа исполнитель меняет статус задачи с «Назначена» на «В работе» или «Отклонена», если задача не в компетенции исполнителя. При необходимости уточнения данных для дальнейшего анализа проблемы статус задачи переводится в «Запрос информации». При решении задачи исполнитель меняет статус на «Выполнена»;

Получение оповещений. Вариант использования «Получение оповещений» исполнителем, так же предполагает возможность разрешить или запретить получение оповещений от системы;

Просмотр оповещений. Данный вариант использования предполагает возможность просмотра полученного оповещения о назначении задач, о получении обратной связи по запросам в рамках задачи, об отмене задачи, о создании подзадач или связанной задачи, об изменении контрольного срока, активации задачи, изменений приоритета задачи, о переназначении и закрытии задачи.

Согласно диаграмме вариантов использования системы руководителем, (рисунок 2), видно, что функция по созданию и назначению задач доступна только руководителю отдела. В свою очередь, исполнитель может изменять статус задач на следующие: «В работе», «Запрос информации», «Отклонена», «Выполнена». Статус задачи «Назначена» проставляется автоматически, когда руководитель назначает задачу. «Закрыта» - это конечный статус проставляется после оценки выполненной задачи.

На рисунке 4 представлена диаграмма деятельности создания задачи.

Диаграмма демонстрирует описание процесса создания задачи. При выявлении нештатной ситуации от пользователей сопровождаемой автоматизированной системы (АС) или от систем мониторинга регистрируется обращение. Обращение проходит первичный анализ, по результатам которого может быть предоставлены решения по типовым ситуациям, или создается задача, которая направляется инженерам сопровождающих АС. Вовремя назначения задачи на сотрудника происходит

отправка оповещения о назначение задачи сотруднику. В зависимости от признака выбранного сотрудником уведомление поступит или нет.

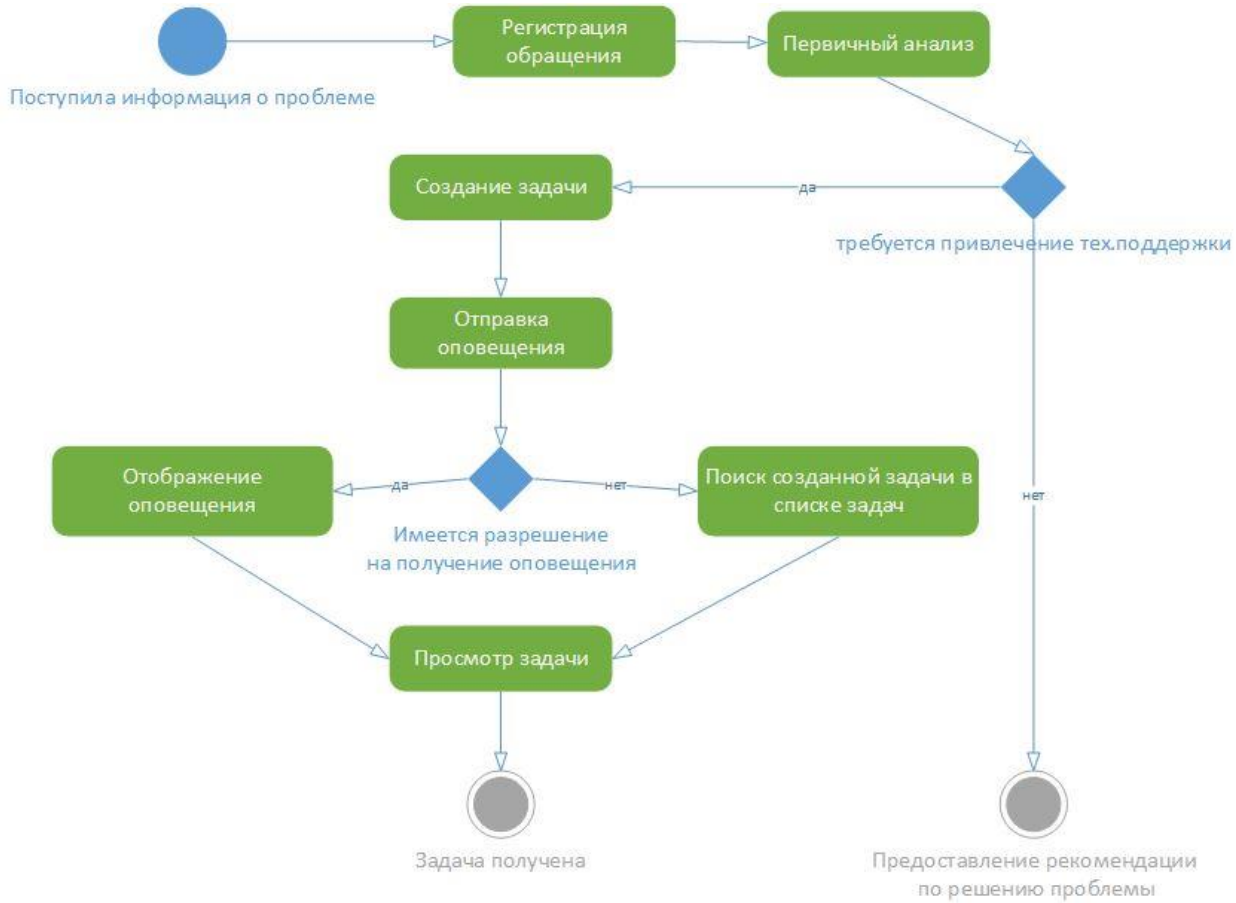

Рисунок 4 – Диаграмма деятельности «Создание задачи»

На рисунке 5 рассмотрена диаграмма деятельности на примере процесса изменения статуса задачи. После получения задачи, исполнитель, приступая к её анализу и решению, меняет статус задачи с «Назначена» на «В работе». О смене статуса происходит отправка оповещения инициатору задачи, что позволяет ему контролировать процесс решения задачи. Во время анализа задачи на основе массового поступления обращений в рамках одной проблемы исполнитель может принять решение о повышении приоритета задачи, что в свою очередь влечет изменение контрольного срока. Об изменении приоритете и контрольного срока инициатору задачи так же отправляется уведомление.

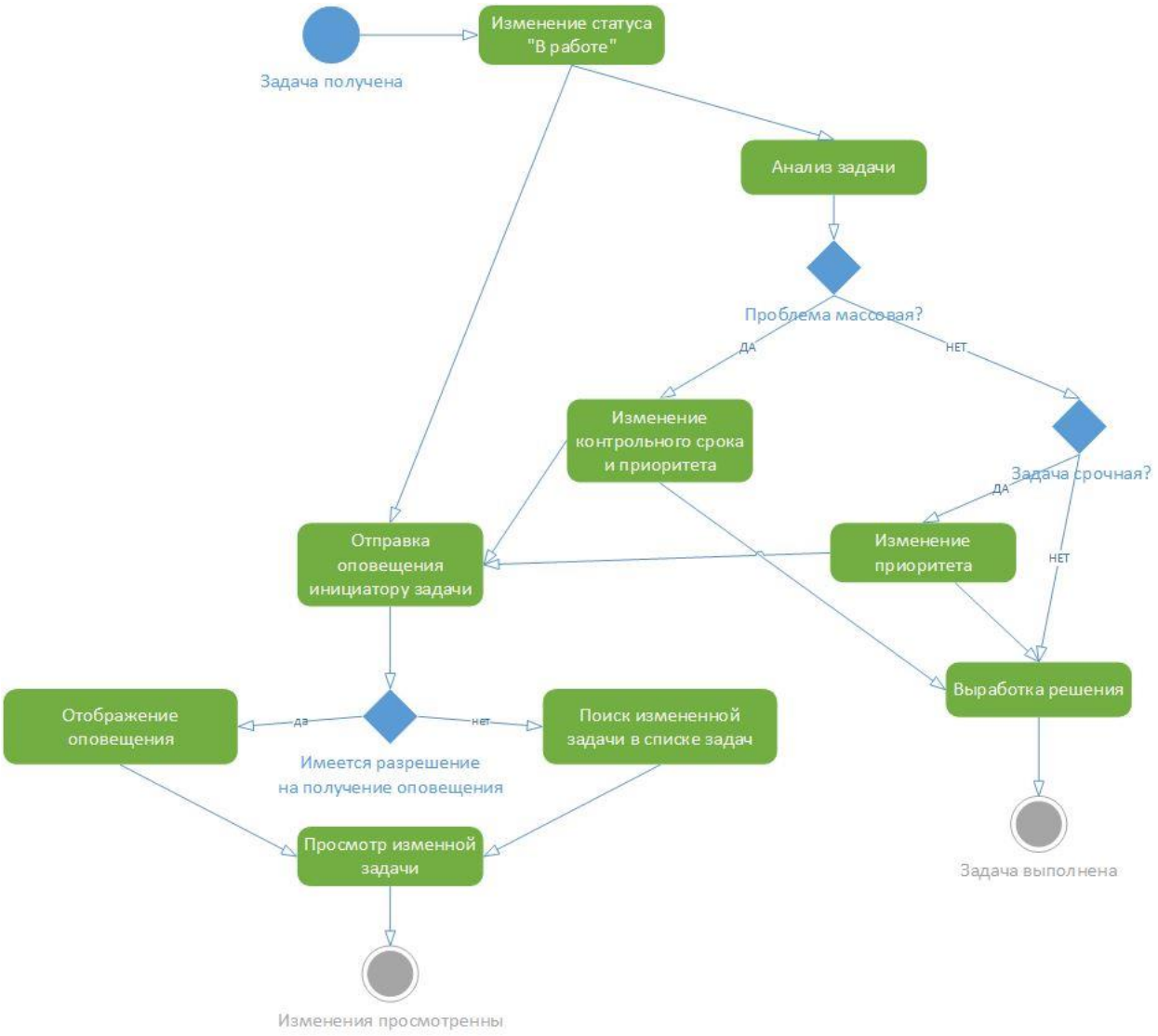

Рисунок 5 – Диаграмма деятельности процесса «Изменение статуса задачи»

На рисунке 6 представлена диаграмма последовательности – регистрация в системе оповещения и работы с задачами.

Пользователь (исполнитель или руководитель) должен заполнить обязательные поля в форме авторизации, отображаемой в браузере. Отсутствие логина или пароля не позволит пройти валидацию. После того, как данные заполнены, происходит формирование и передача данных POST-запросом на веб-сервер, а затем устанавливается соединение с сервером базы данных (БД) и передача данных. Происходит поиск аккаунта, и затем передача результата. После успешной регистрации, пользователю будет отображена

страница авторизации. Если введенные данные некорректны, то сотруднику отображается соответствующее сообщение. Важно отметить, что в системе могут зарегистрироваться сотрудники, данные о которых уже имеются в базе данных, иначе говоря, только сотрудники, работающие в организации. При успешной авторизации сотруднику будет отображена страница навигации.

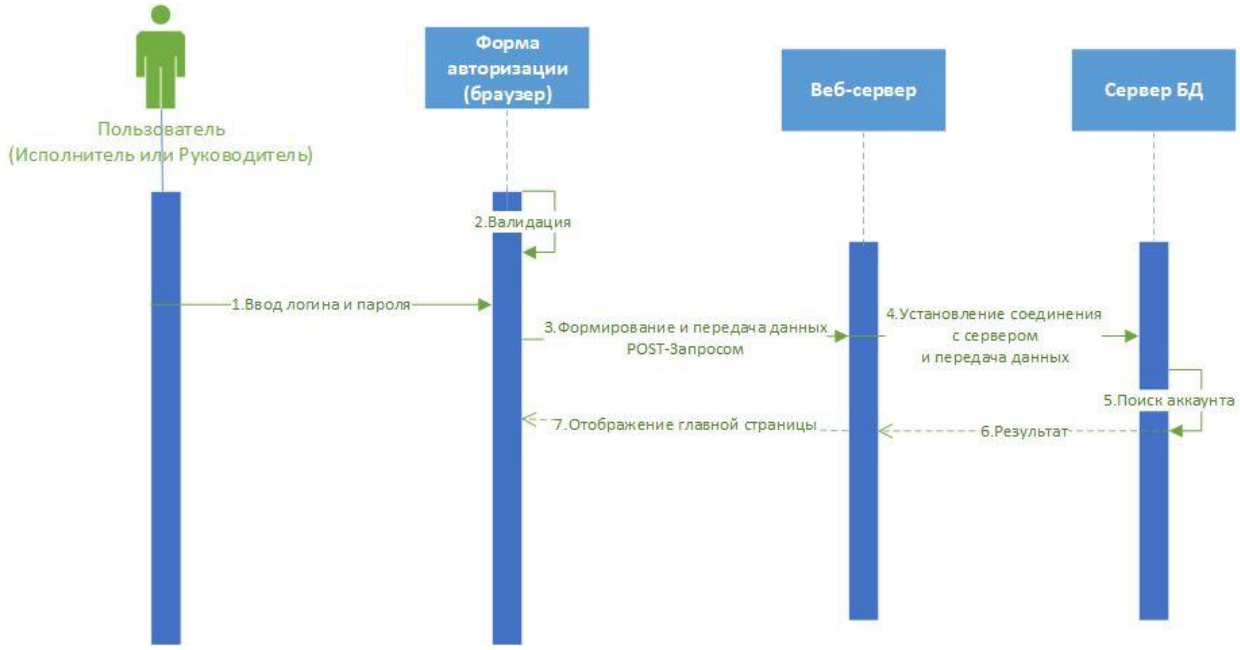

Рисунок 6 - Диаграмма последовательности «Регистрация в системе оповещения и работы с задачами»

На рисунке 7 представлена диаграмма последовательности – создания задачи.

Руководитель вводит данные по задаче, происходит проверка, заполнены ли все обязательные поля. При корректном заполнении осуществляется формирование и передача данных POST-запросов на веб-сервер, а затем устанавливается соединение с сервером БД и передача данных. На сервере БД происходит сохранение задачи и передача результата на Веб-сервер. На Веб-сервере происходит формирование и передача данных POST-запросом на сервер push-уведомлений, для формирования и отправки push-уведомления пользователю.

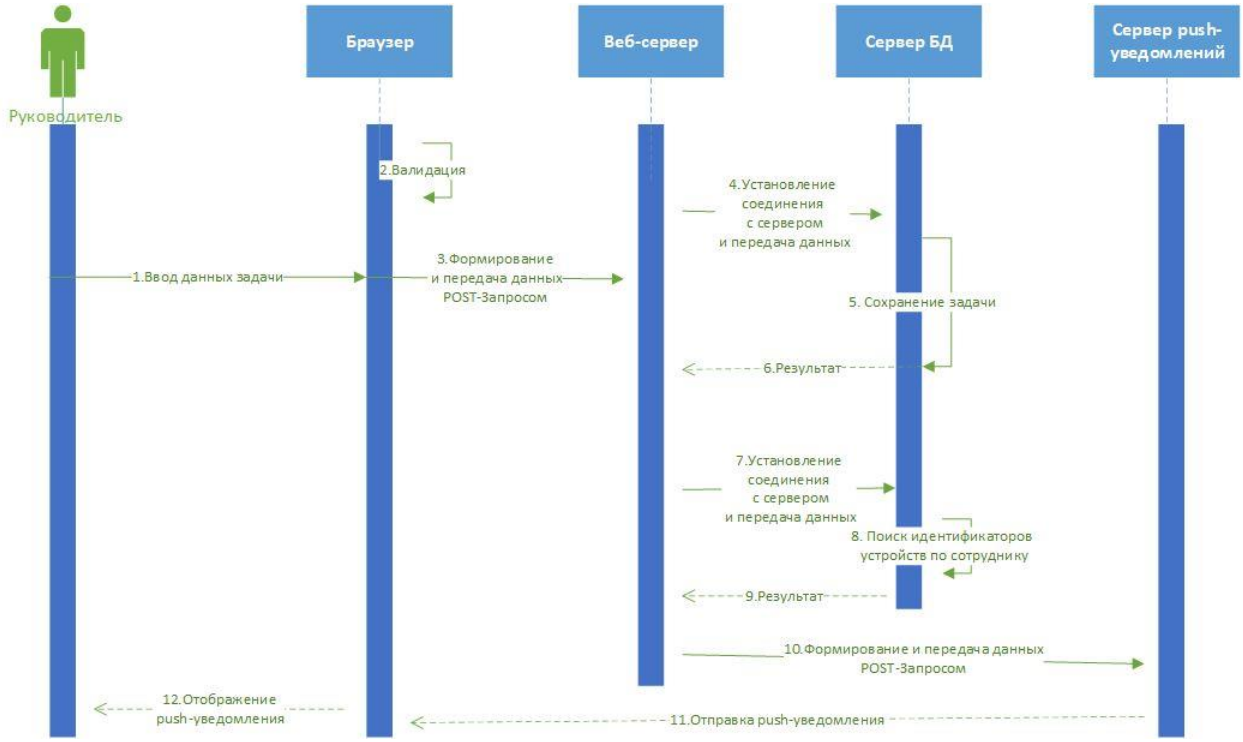

Рисунок 7 - Диаграмма последовательности «Создание задачи»

2.2 Логическое проектирование базы данных

Проектирование базы данных - это один из основных этапов при разработке приложения для оповещения и управления задачами сотрудников.

В рамках логического проектирования были выделены следующие сущности: сотрудник, отдел, пользователь, задача, роль пользователя, устройство. Данные сущности и их атрибуты представлены на рисунке 8.

Таблица «Сотрудник» хранит информацию о логине, имени, фамилии, должности, отделе и начальнике. Идентификатор руководителя сотрудника ссылается на кортеж из таблицы сотрудников. Идентификатор отдела ссылается на номер отдела, который является первичным ключом в таблице отделов. Атрибут «имя пользователя» также является внешним ключом, ссылающимся на кортеж в таблице пользователей. Таблица сотрудников связана с таблицей пользователей, отношение «один-к-одному» Таблица сотрудников связана с таблицей отделов отношением «один-ко-многим».

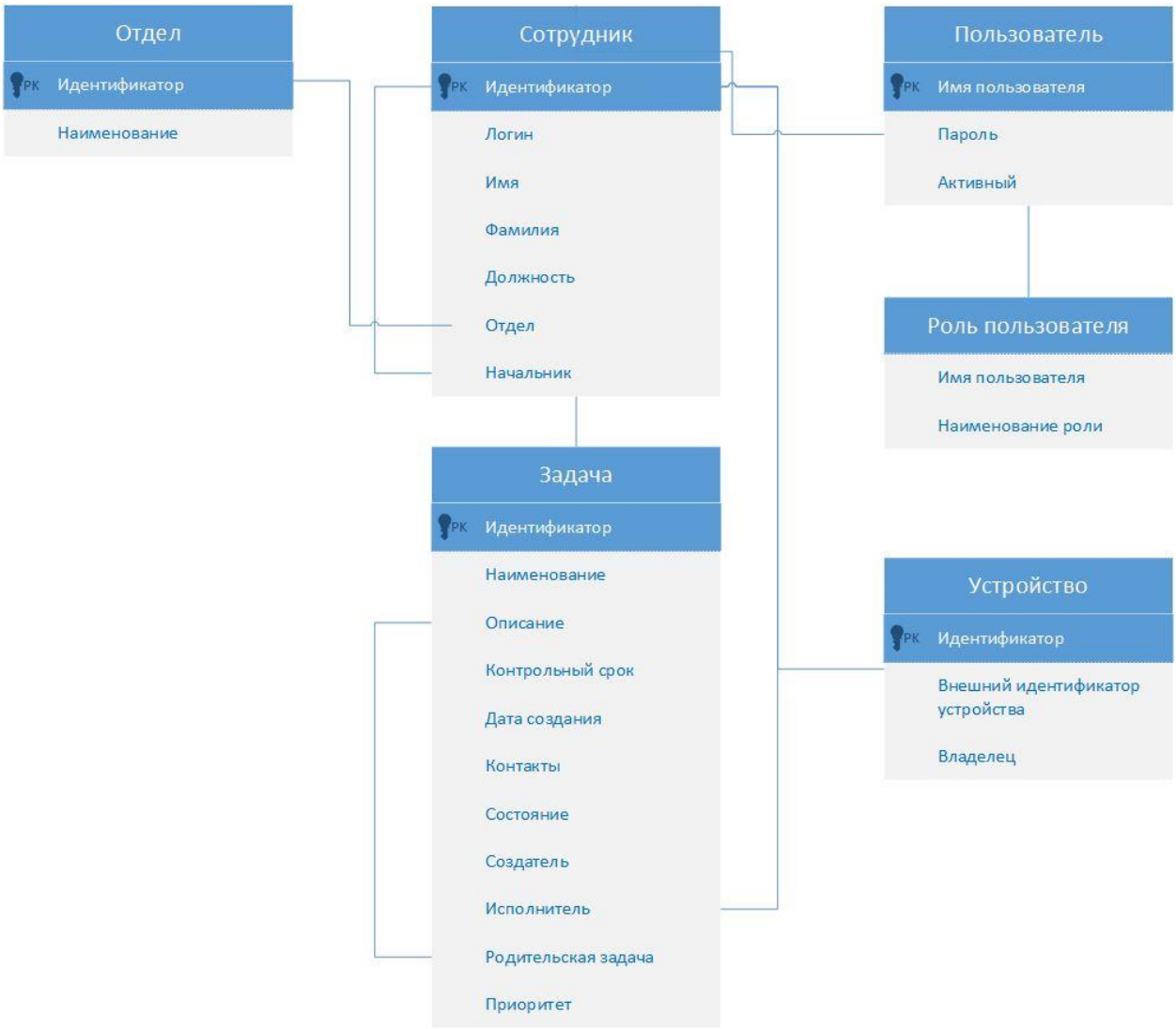

Рисунок 8 – Модель базы данных

Таблица «Пользователь» хранит информацию о зарегистрированных пользователях. В таблице пользователей хранится информация о зарегистрированных пользователях. К такой информации относятся имя зарегистрированного пользователя, которое является первичным ключом.

Таблица «Роль пользователя» хранит информацию о ролях зарегистрированных пользователей. Данная таблица связана с таблицей пользователей отношение один-ко-многим, т.е. один пользователь может иметь несколько ролей. Первичным ключом является идентификатор, значение которого генерируется при регистрации пользователя. Также в

таблице содержится информация об имени пользователя и доступных ему ролях.

Таблица «Отдел» хранит информацию об отделах предприятия и включает в себя информацию о номерах отделов и название отдела.

Таблица «Устройство» содержит информацию об идентификаторе устройства, о строковом идентификаторе устройства, который выдается сервером уведомлений, и идентификаторе сотрудника, который является вторичным ключом и ссылается на кортеж и таблицы сотрудников. Идентификатор является первичным ключом, а его значение берется из последовательности при добавлении устройства.

Таблица «Задача» хранит информацию о задачах сотрудников. В таблице видно, что задача имеет идентификатор, который является первичным ключом. Помимо этого, задача обладает именем, которое не может пустым, описанием, датами выполнения и создания, приоритетом, контактами, текущим статусом. Задача имеет исполнителя, создателя и родительскую задачу. Все три атрибута являются вторичными ключами. Идентификаторы создателя и исполнителя ссылаются на соответствующие кортежи из таблицы сотрудников. Каждый из них связан с таблицей сотрудников отношение один-ко-многим. Идентификатор родительской задачи ссылается на кортеж из таблицы задач. Задача может и не иметь ссылку на родительскую задачу. Описание задачи и контакты также являются опциональными атрибутами в отличие от имени, даты выполнения, даты создания, идентификаторов исполнителя и создателя, приоритета, текущего статуса.

2.3 Определение технологий и средств разработки приложения

Далее проведем анализ и определим набор инструментов и технологий, позволяющих реализовать каждый из перечисленных пунктов.

Для разработки веб-приложений будет использован язык программирования Java. Существуют различные языки программирования, с

помощью которых возможно реализовать серверную часть веб-приложения. К таким языкам относятся JavaScript, Python, PHP, Java, C#, Ruby. Для реализации серверной части системы управления задач и системы оповещения был выбран язык Java по следующим причинам:

– Многопоточность: многопоточная обработка HTTP-запросов от клиентов позволяет ускорить работу приложения.

– Безопасность: языка Java достигается за счет большого набора API, инструментов и реализаций широко используемых алгоритмов, механизмов и протоколов безопасности. API-интерфейсы Java включают криптографию, инфраструктуру открытых ключей, безопасную связь, аутентификацию и контроль доступа. JDK изначально разработан с упором на безопасность, что позволяет создавать безопасные приложения с самого начала.

– Библиотеки и фреймворки: существует большое количество Java-библиотек и фреймворков различного назначения, использование которых значительно ускоряет программирование серверной части веб-приложений.

– Масштабируемость: автоматическое управление памятью и сборка мусора обеспечивают высокую масштабируемость Java и ускоряют разработку веб-приложений.

Программы, написанные на Java, являются переносимыми. Свойство переносимости заключается в том, что приложения должны выполняться на любой совокупности аппаратного обеспечения и операционной системы. Такой эффект достигается за счет компиляции кода языка Java в промежуточное представление под названием байт-код, а не непосредственно в машинный код конкретной архитектуры [2][11].

Как и локальные приложения, веб-приложения состоят из многих частей, и часто содержат компоненты, некоторые из которых имеют пользовательские интерфейсы, а некоторые из них не требуют графического интерфейса пользователя (GUI). Кроме того, для реализации графического интерфейса веб-приложения часто требуется использование языков разметки,

стилей или скриптового языка, такие как HTML, CSS, или языка программирования JavaScript.

Платформа Java EE предоставляет возможность для создания веб-приложения с использованием технологий Java, например, таких как Java сервлет и JSP [1]. Однако на данный момент JSP является устаревшим инструментом и не существует инструмент, который стремится стать полной заменой JSP для создания пользовательского интерфейса для Java приложений. Неким аналогом является шаблонизатор Thymeleaf, который использует концепцию “естественных шаблонов”, чтобы внедрить свою логику в файлы шаблонов таким образом, чтобы этот шаблон не влиял на отображение прототипа дизайна [18] [19]. Данный инструмент используется для реализации пользовательского интерфейса веб-приложения в текущем бакалаврской работе. Thymeleaf-шаблон представляет собой HTML-документ, в котором используются как стандартные HTML-теги, так и специальные HTML-теги, предоставляемые шаблонизатором.

Типичное корпоративное веб-приложение, которым является разрабатываемая система управления задач и система оповещения, состоит из нескольких слоев, как показано на рисунке 9.

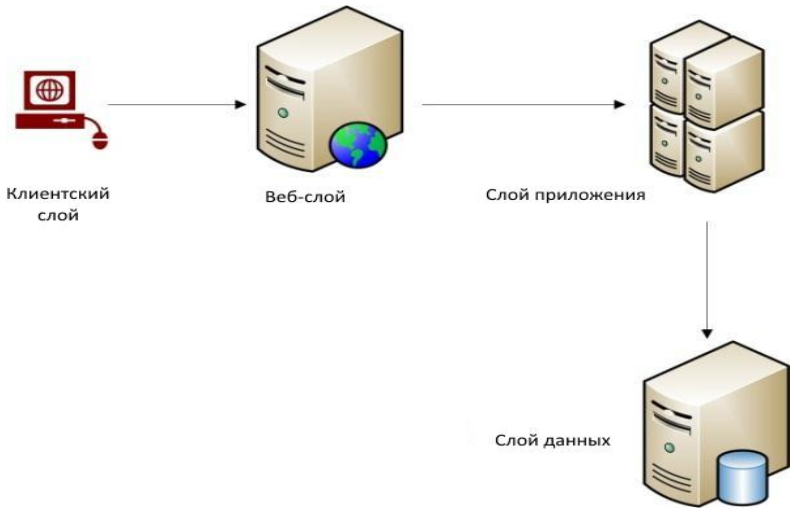

Рисунок 9 - Многоуровневая архитектура системы управления задач и системы оповещения

Важно отметить, что данное разделение является логическим, физически все слои могут быть развернуты в рамках одной или нескольких систем.

Далее, кратко рассмотрим, как все эти слои взаимодействуют друг с другом для выполнения задач веб-приложения предприятия.

Первый слой данных хранит данные предприятия и предоставляет функции для работы с ними. Он должен быть размещен на конкретной платформе баз данных, например, таких как MySQL, SQL Server, Oracle. Проблема выбора платформы базы данных носит многогранный характер. Нужно учитывать нуждается ли приложение в поддержке нескольких платформ баз данных, сильные стороны конкретной платформы базы данных и связанные с ними затраты на разработку и техническое обслуживание и т.д. Для реализации была выбрана СУБД PostgresQL.

Второй слой приложения реализует бизнес-логику. Для поддержки выполнения бизнес-логики требуется среда выполнения. Такой средой может быть один из серверов приложений продуктов, построенных на Java технологиях.

Третий веб-слой. Базовая функция веб-слоя заключается в получении сервисом запросов, делегации таких запросов слою приложения и отправка результатов обработки обратно к пользователям. Для крупных корпоративных веб-приложений, веб-слой и слой приложения могут быть разделены и размещены на двух или более отдельных физических системах или кластерах.

Четвертый клиентский слой - это взаимодействие пользователя с веб-приложением предприятия, в основном через протоколы HTTP или HTTPS. На этом уровне ответы от веб-сервера предоставляются пользователям через приложения, такие как браузеры и т.д. [1]. Далее проанализируем технологии и средства для разработки веб-приложения.

Для разработки приложения на Java, в том числе и веб-приложения, необходим комплект разработчика Java (JDK). Он включает в себя компилятор Java, набор стандартных библиотек, документацию. Для написания программного кода используется интегрированная среда разработки (IDE). В

качестве такой среды была выбрана IntelliJ IDEA 2021, предоставляемая и разрабатываемая компанией JetBrains, поскольку данная среда разработки является лидером среди IDE, используемых для разработки Java приложений (Eclipse, NetBeans). IntelliJ IDEA Ultimate 2021 поддерживает достаточное количество языков программирования, в том и числе и необходимые для разработки веб-приложения, имеет широкий функционал и позволяет быстро и удобно создавать различные программные продукты.

Для реализации приложения по управлению задачами и оповещения пользователей нам необходимо спроектировать и создать базу данных. Приложение во время функционирования оперирует с данными, которые хранятся в оперативной памяти устройства. После того, как приложение завершило свою работу используемые данные нигде не сохраняются. Чаще всего требуется постоянное хранение данных. Для этого применяются базы данных. Базой данных называется совокупность взаимосвязанных данных, организованных в соответствии с формальным описанием данных таким образом, чтобы с ними мог работать пользователь. Для управления базами данных используются системы управления базам данных, далее СУБД. СУБД представляет собой совокупность программных и языковых средств, обеспечивающих управление базами данных.

Для разрабатываемого приложения будет использована СУБД PostgresQL. PostgresQL представляет собой объектно-реляционную систему управления базами данных. Так как Java-приложения оперируют объектами, то для взаимодействия с базами данных удобно использовать ORM-подход. Объектно-реляционное отображение (object-relational mapping, ORM) представляет собой технологию программирования для связывания базы данных с концепциями объектно-ориентированных языков программирования, которым и является Java. Для Java возможности объектно-ориентированного отображения предоставляет программный интерфейс Java Persistence API (JPA). Существует несколько реализаций данного интерфейса,

одну из таких реализаций представляет Hibernate, которая и была выбрана для разрабатываемого приложения [13].

Для запуска приложения на компьютере необходимо осуществить процесс преобразования файлов с исходным кодом в отдельный артефакт, который и будет впоследствии запущен. Такой процесс называется сборкой приложения. Конечно, для простых приложений сборку можно осуществлять вручную, но для более крупных и сложных проектов данный способ не является оптимальным. Поэтому используются средства автоматизации сборки приложений. Одним из средств, осуществляющих автоматизацию сборки приложений, является Apache Maven. Maven представляет собой фреймворк, который помогает управлять проектами Java, а также автоматизировать сборку приложения. Для автоматизации сборки проекта, содержащий исходный код системы управления задач и системы оповещения, будет использован фреймворк Maven.

Так как многие задачи разработки веб-приложения могут повторяться или требовать наличие избыточного шаблонного кода, то для уменьшения количества общих действий следует применять фреймворки. Такими фреймворками являются Spring Framework и связанный с ним Spring Boot Framework. Spring Framework представляет собой универсальный фреймворк с открытым исходным кодом. Он обеспечивает комплексную модель разработки и конфигурации современных бизнес-приложений для Java-платформы. Несмотря на свою масштабность, данный фреймворк легок в использовании, так как Spring имеет модульную конструкцию, что в свою очередь позволяет использовать только те части, которые необходимы.

Для разработки системы управления задач и системы оповещения были выбраны следующие модули фреймворка Spring:

– основной контейнер Spring;

– доступ к данным и интеграции: JDBC, Transaction, ORM для организации доступа серверной части веб-приложения к СУБД;

– веб и удаленное взаимодействие: spring-web и spring-webmvc для реализации серверной части по обработке HTTP-запросов от клиентской части веб-приложения.

Spring Boot фреймворк упрощает создание автономных приложений производственного уровня на основе фреймворка Spring, которые можно «просто запустить». Данный фреймворк был создан для того, чтобы упростить настройку фреймворка Spring. Перечисленный инструментарий может быть использован для создания системы управления задачами сотрудников. Так как система оповещения реализуется для веб-приложения с помощью браузерных push-уведомлений, то необходимо использовать технологии, предназначенные для работы с веб-приложением и браузером. Для разрабатываемой системы оповещения были использованы реализации спецификаций Push API и Notification API, предоставляемых браузером Chrome. Chrome использует FCM (Firebase Cloud Messaging) для обработки отправки и доставки push сообщений. Но для использования этого сервиса необходимо зарегистрироваться в консоли разработчика, создать проект и подключить к нему FCM API. После создания проекту присваивается идентификатор, или иначе номер. После подключения сервиса Firebase выдает ключ на его пользование. Push API в Chrome опирается на несколько различных частей технологии. К ним можно отнести манифест веб-приложения и Service Worker. Манифест веб-приложения представляет собой файл, основанный на формате JSON, который предоставляет разработчикам централизованное место для размещения метаданных, связанных с веб-приложением. Эти метаданные включают, но, не ограничиваясь этим, имя веб-приложения, ссылки на иконки, а также предпочтительную ссылку, которая будет открываться при запуске приложения. Манифест также позволяет разработчикам указывать ориентацию по умолчанию для их веб-приложения, а также предоставляет возможность установить режим отображения для приложения (например, полноэкранный режим) [7]. Важным параметром для реализации системы push-уведомлений является gcm_sender_id, который

принимает значение номер проекта, созданного в консоли разработчика Firebase. Данный параметр используется браузером Chrome, когда FCM осуществляет выдачу идентификатора устройству пользователя. Это означает, что FCM может идентифицировать пользовательское устройство и убедиться, что номер проекта соответствует выданному ключу на пользование API на сервере и что пользователь разрешил принятие сообщения провайдера. Ключ API используется FCM, чтобы найти подходящий номер проекта и удостовериться, что пользователь дал разрешение на уведомления для созданного в консоли проекта и, наконец, гарантируя, что IP-адрес сервера входит в список разрешенных серверов для этого проекта. Service Worker представляет собой скрипт, который запускается в браузере в фоновом режиме, отдельно от веб-страницы, открывая дверь к функциям, которые не нуждаются в веб-странице или взаимодействии с пользователем. Одной из областей применения данного скрипта являются браузерные push-уведомления [14].

Выводы по главе 2

Проведя анализ средств, применение которых возможно для разработки системы оповещения пользователя сети, мы определились с технологиями и инструментами разработки веб-приложения. Реализацию бизнес-логики веб-приложений будем осуществлять с использованием объектно-ориентированного языка Java. Выбор языка программирования Java для реализации серверной части веб-приложения связан с основными преимуществами, а именно многопоточность, безопасность, наличие большого количество Java-библиотек и фреймворков различного назначения, а также масштабируемостью. Для реализации клиентской части веб-приложения выбран шаблонизатор Thymeleaf, который реализует концепцию «естественных шаблонов», благодаря чему шаблон представляет собой читабельную HTML-страницу. Для стилизации графического интерфейса,

представляемого набором HTML-страниц, выбран язык CSS. Для реализации отправки запросов с клиента на сервер выбран язык JavaScript.

Для разработки Java-приложений необходим пакет разработчика Java и интегрированная среда разработки (IDE). В качестве такой среды была выбрана IntelliJ IDEA 2021.

В качестве основы реализации веб-приложения нами использован шаблон проектирования Model-View-Controller (MVC). Шаблоны проектирования MVC позволяют ускорить процесс разработки, предоставляя уже проверенные решения. Паттерн MVC представляет собой совокупность шаблонов проектирования. Основная идея такого паттерна заключается в разделении модели данных приложения, пользовательского интерфейса и логики приложения на три отдельных компонента, так, чтобы изменения одного компонента минимально сказывались на остальных. Реализацию данного паттерна для веб-приложений содержит фреймворк Spring. Фреймворк Spring также способствует уменьшению шаблонного кода и упрощению разработки сложных и крупных приложений. Spring представляет собой универсальный фреймворк для Java-платформы.

Разрабатываемое приложение для управления задачами и оповещениями является клиент-серверным приложением. Обмен данными между клиентом приложения и сервером осуществляется посредством протокола HTTP. Помимо взаимодействия клиента с сервером приложений, осуществляется взаимодействие между сервером приложений и сервером уведомлений. Общение между двумя данными серверами также осуществляется с помощью протокола HTTP. Данный протокол функционирует по схеме “запрос-ответ”.

Для организации хранения данных нами будет использоваться реляционная база данных, модели и структура которой будет описана в следующей главе. Для передачи данных в текстовом формате по сети используется универсальный формат JSON.

Глава 3 Реализация системы оповещения пользователей сети

3.1 Проектирование приложения управления задачами и оповещения сотрудников

Разрабатываемое приложение представляет собой систему управления задачами и оповещения сотрудников. Для обеспечения корректного функционирования приложения требуется постоянное хранение данных. Для достижения этой цели была использована объектно-реляционная СУБД PostgresQL. Данная СУБД была выбрана среди Oracle, MySQL, Microsoft SQL server, PostgresQL, поскольку обладает следующими преимуществами:

– поддержка БД неограниченного размера;
– мощные и надежные механизмы транзакций и репликации;
– легкая расширяемость.

Так как для хранения данных используется реляционная база данных, представляющая собой набор взаимосвязанных таблиц, то была спроектирована структура таблиц базы данных (рисунок 10). Скрипт создания таблиц базы данных приведен в Приложении А.

Сотрудник имеет идентификатор. Идентификатор является первичным ключом. В таблице также присутствуют атрибуты «имя», «фамилия» и «должность». Помимо ранее перечисленных атрибутов в таблице сотрудников в отдельных колонках хранятся идентификаторы отдела, к которому сотрудник относится, и идентификатор начальника. Идентификатор менеджера сотрудника ссылается на кортеж из таблицы сотрудников. Идентификатор отдела ссылается на номер отдела, который является первичным ключом в таблице отделов. Атрибут «имя пользователя» также является внешним ключом, ссылающимся на кортеж в таблице пользователей. Таблица сотрудников связана с таблицей пользователей, отношение «один-к-одному» Таблица сотрудников связана с таблицей отделов отношением «один-ко-многим».

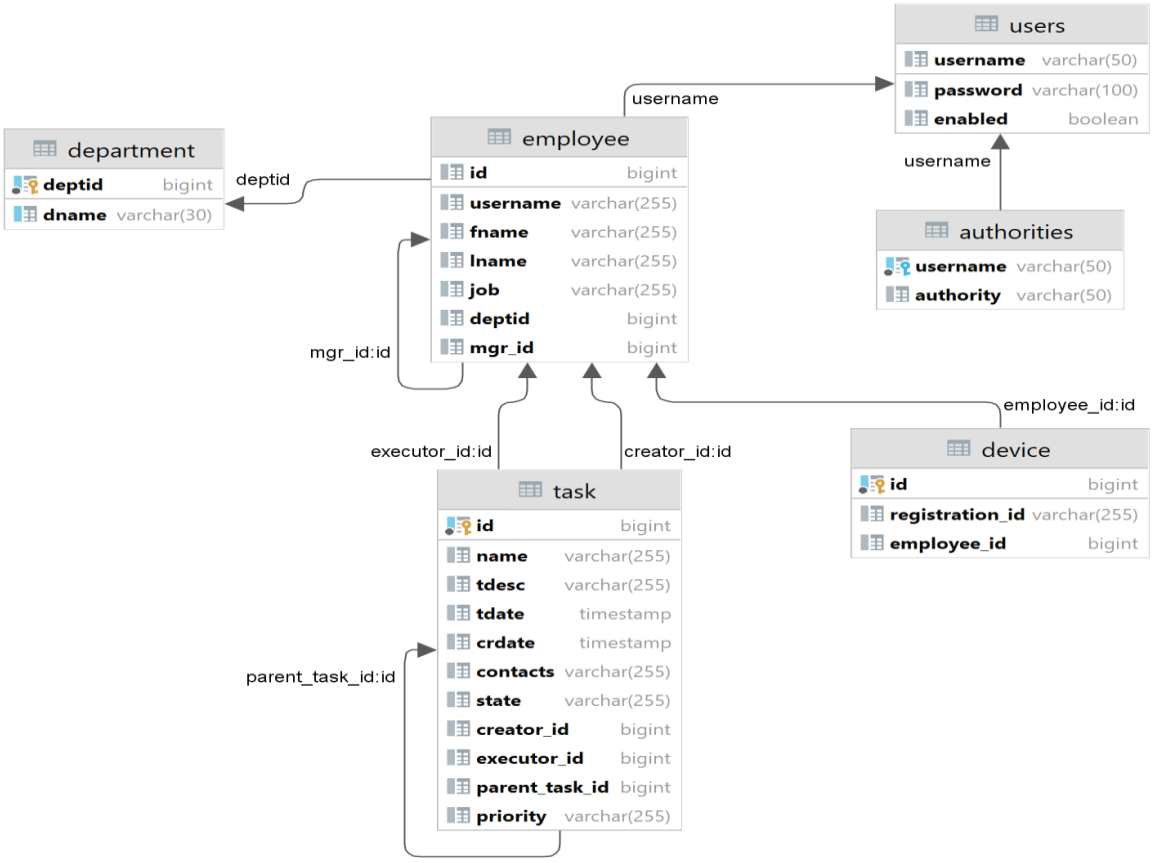

Рисунок 10 - Структура базы данных

Далее подробно рассмотрим назначение каждой из таблиц и дадим описание каждому из атрибутов таблиц. Таблица EMPLOYEE (таблица 4) хранит информацию о сотрудниках.

Таблица 4 - Описание атрибутов таблицы EMPLOYEE

Имя	Описание
ID	Идентификатор сотрудника
FNAME	Имя сотрудника
LNAME	Фамилия сотрудника
JOB	Должность сотрудника
MGR_ID	Идентификатор начальника сотрудника, является ключом к строке начальника в этой же таблице.
USERNAME	Имя пользователя, под которым сотрудник зарегистрирован, является ключом к строке в таблице USERS.
DEPTID	Идентификатор отдела, в котором работает сотрудник. Представляет собой ключ, который ссылается на строку в таблице DEPARTMENT.

Таблица USERS (таблица 5) хранит информацию о зарегистрированных пользователях.

Таблица 5 - Описание атрибутов таблицы USERS

Имя	Описание
USERNAME	Имя зарегистрированного пользователя. Является уникальным.
PASSWORD	Пароль зарегистрированного пользователя. Хранится не сам пароль, а его хэш-сумма
ENABLED	Указывает, разрешен ли пользователь или заблокирован.

Как видно из таблицы 5, в таблице пользователей хранится информация о зарегистрированных пользователях. К такой информации относятся имя зарегистрированного пользователя, которое является первичным ключом в данной таблице, хэш-сумма пароля пользователя и указатель, принимающий значение 0 или 1, который говорит, разрешен ли пользователь или заблокирован. Заблокированный пользователь не может пройти проверку подлинности. Имя пользователя должно быть уникальным.

Таблица AUTHORITIES (таблица 6) хранит информацию о ролях зарегистрированных пользователей.

Таблица 6 - Описание атрибутов таблицы AUTHORITIES

Имя	Описание
USERNAME	Имя пользователя, является ключом к строке из таблицы USERS.
AUTHORITY	Роль пользователя, принимает значения {ROLE_USER, ROLE_ADMIN}

В таблице 6 отображена информация о ролях пользователей. Данная таблица связана с таблицей пользователей отношение один-ко-многим, т.е. один пользователь может иметь несколько ролей. Первичным ключом является идентификатор, значение которого генерируется при регистрации

пользователя. Также в таблице содержится информация об имени пользователя и доступных ему ролях.

Таблица DEPARTMENT (таблица 7) хранит информацию об отделах предприятия.

Таблица 7 - Описание атрибутов таблицы DEPARTMENT

Имя	Описание
DEPTID	Идентификатор отдела.
DNAME	Имя отдела, является уникальным.

Таблица 7 включает в себя информацию о номерах отделов и название отдела. Номер является первичным ключом, название отдела должно быть уникальными.

Как видно из таблицы 8, таблица устройств содержит информацию об идентификаторе устройства, о строковом идентификаторе устройства, который выдается сервером уведомлений, и идентификаторе сотрудника, который является вторичным ключом и ссылается на кортеж и таблицы сотрудников. Идентификатор является первичным ключом, а его значение берется из последовательности при добавлении устройства.

Таблица DEVICE (таблица 8) хранит информацию об устройствах сотрудников.

Таблица 8 - Описание атрибутов таблицы DEVICE

Имя	Описание
ID	Идентификатор устройства, генерируется последовательностью REG_ID_SEQ
REGISTRATION_ID	Строковый идентификатор устройства, присваиваемый сервером уведомлений.
EMPLOYEE_ID	Идентификатор сотрудника, который заходил с устройства, является ключом к строке из таблицы EMPLOYEE

Таблица TASK (таблица 9) хранит информацию о задачах сотрудников. В таблице 9 видно, что задача имеет идентификатор, который является первичным ключом. Помимо этого, задача обладает именем, которое не может пустым, описанием, датами выполнения и создания, приоритетом, контактами, текущим статусом. Стоит отметить, что дата создания по умолчанию принимает значение текущей даты. Приоритет принимает значение из множества {0, 1, 2}. 0 – Низкий приоритет, 1 – обычный приоритет, 2 – высокий приоритет. Задача может иметь один из выделенных статусов, т.е. принимает значение {N, A, P, C}, где N – новая задача, A – завершенная задача, P – задача в процессе выполнения, C – отмененная задача. Помимо перечисленной информации, задача имеет исполнителя, создателя и родительскую задачу. Все три атрибута являются вторичными ключами.

Таблица 9 - Описание атрибутов таблицы TASK

Имя	Описание
ID	Идентификатор задачи, генерируется последовательностью TASK_ID_SEQ
NAME	Имя задачи, не может быть пустым
TDESC	Описание задачи
TDATE	Дата выполнения задачи
CRDATE	Дата создания задачи, по умолчанию принимает значение текущей даты
PRIORITY	Приоритет задачи, принимает значения {0,1,2}.
CONTACTS	Контакты задачи
STATE	Статус задачи, может принимать значения {N, A, P, C}
CREATOR_ID	Идентификатор создателя задачи, является ключом, который ссылается на строку в таблице EMPLOYEE.
EXECUTOR_ID	Идентификатор исполнителя задачи, является ключом, который ссылается на строку в таблице EMPLOYEE.
PARENT_TASK_ID	Идентификатор родительской задачи, является ключом, который ссылается на строку в этой же таблице.

Идентификаторы создателя и исполнителя ссылаются на соответствующие кортежи из таблицы сотрудников. Каждый из них связан с таблицей сотрудников отношение один-ко-многим. Идентификатор

родительской задачи ссылается на кортеж из таблицы задач. Задача может и не иметь ссылку на родительскую задачу. Описание задачи и контакты также являются опциональными атрибутами в отличие от имени, даты выполнения, даты создания, идентификаторов исполнителя и создателя, приоритета, текущего статуса.

Разрабатываемое приложение состоит из классов, осуществляющих логику работы приложения, страниц, отображаемых пользователю, файлов с таблицами стилей страниц, а также файлов, осуществляющих обработку действий пользователя в браузере и конфигурации Spring Security.

Взаимосвязь классов, из которых состоит приложение, можно описать с помощью диаграммы классов. Разделим общую диаграмму классов на две части. Первая часть отображает классы, которые связаны с сущностью задачи, осуществляют работу с задачами (рисунок 11).

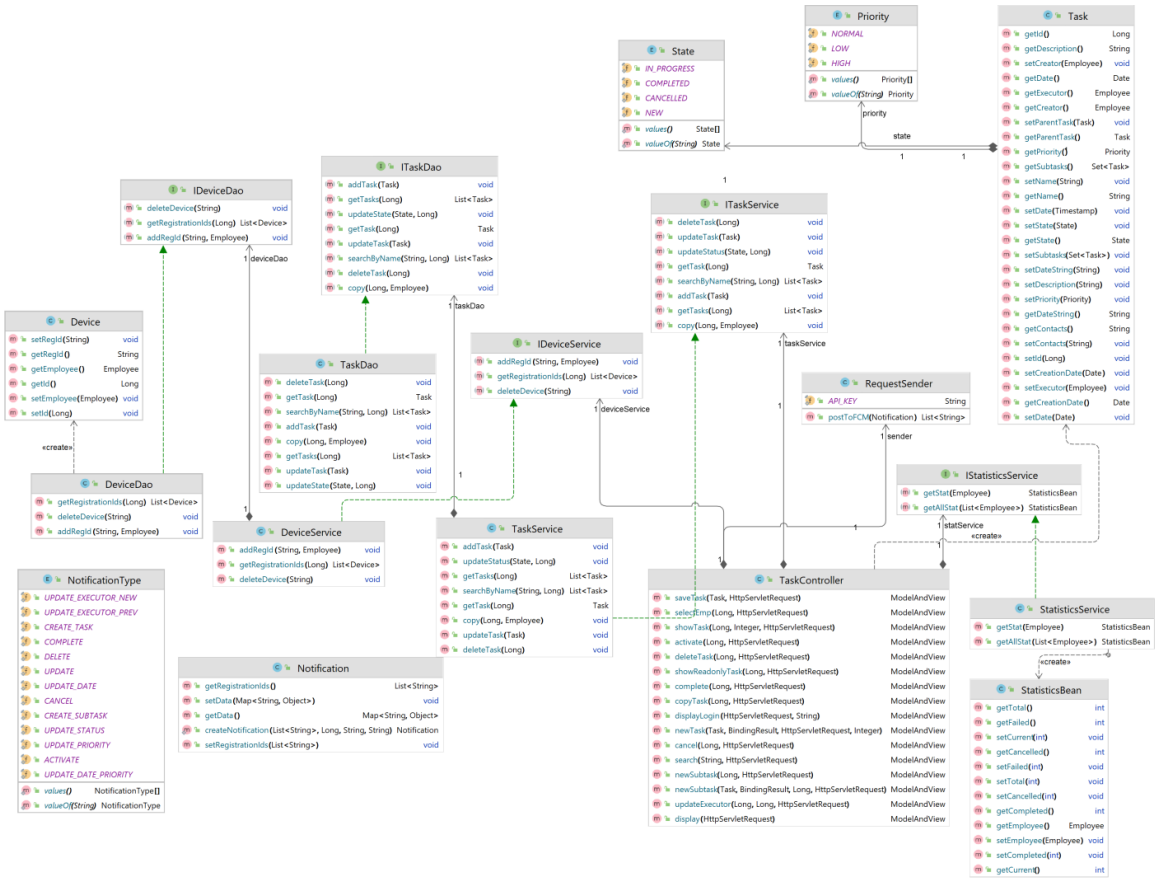

Рисунок 11 - Диаграмма классов (часть 1)

Вторая же часть взаимодействует с сущностью сотрудника, т.е. осуществляет действия, связанные с авторизацией, регистрацией и получением данных, связанных с сотрудником (рисунок 12).

Рисунок 12 - Диаграмма классов (часть 2)

Общая структура классов в системе представлена на рисунке 13.

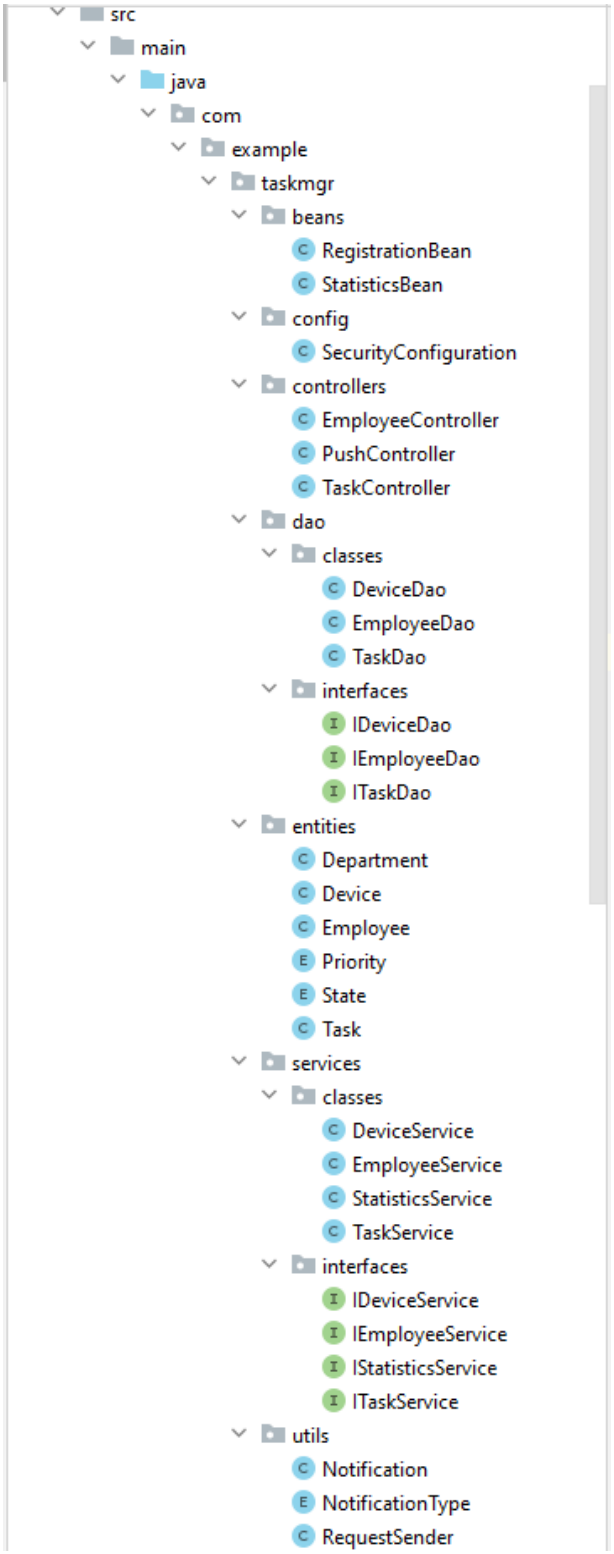

Рисунок 13 - Структура классов

Структура файлов, состоящая из файлов страниц, таблиц стилей, изображений и класса конфигурации Spring Security отображена на рисунке 14.

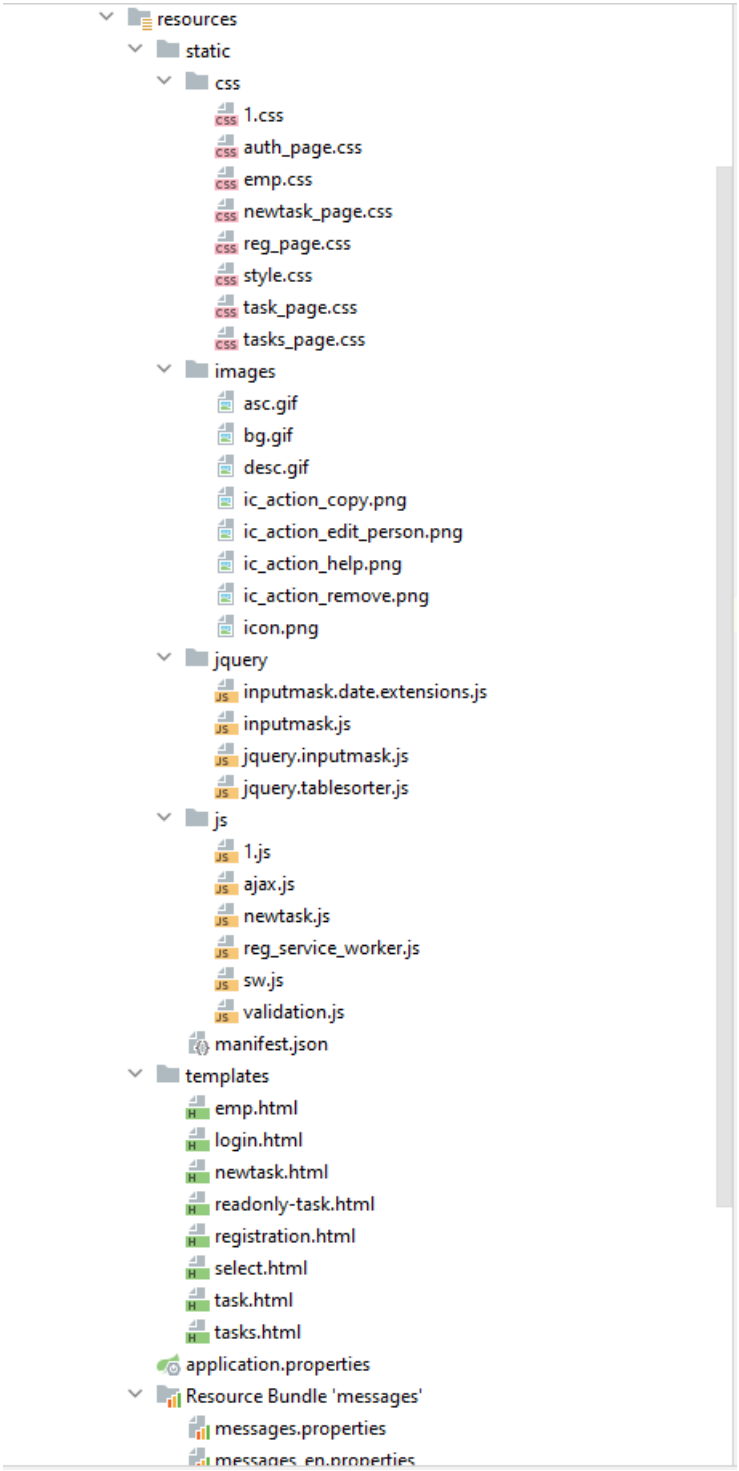

Рисунок 14 - Структура файлов

Далее дадим краткое описание каждого из отображенных классов в таблице 10.

Таблица 10 - Краткое описание классов приложения

Имя класса	Краткое описание
Task	Класс задачи, представляет собой отображение таблицы задач из базы данных
Employee	Класс сотрудника, представляет собой отображение таблицы сотрудников из базы данных.
Department	Класс отдела, представляет собой отображение таблицы отделов из базы данных.
Device	Класс устройства, представляет собой отображение таблицы устройств из базы данных.
TaskController	Осуществляет обработку запросов, связанных с добавлением, удалением, изменением, копированием и просмотром задач, также формирует сообщения для уведомления
PushController	Реализует обработку запросов на уведомление.
EmployeeController	Обрабатывает запросы, связанные с авторизацией, регистрацией сотрудников, а также просмотром информации о сотруднике.
IDeviceDao	Представляет собой интерфейс, содержащий методы для работы с таблицей устройств.
DeviceDao	Реализация интерфейса IDeviceDao.
IEmployeeDao	Представляет собой интерфейс, содержащий методы для работы с таблицей сотрудников.
EmployeeDao	Реализация интерфейса IEmployeeDao.
ITaskDao	Представляет собой интерфейс, содержащий методы для работы с таблицей задач.
TaskDao	Реализация интерфейса ITaskDao.
IDeviceService	Интерфейс сервиса, осуществляющий связь контроллера с интерфейсом IDeviceDao.
DeviceService	Реализация интерфейса IDeviceService.
IEmployeeService	Интерфейс сервиса, осуществляющий связь класса EmployeeController с интерфейсом IEmployeeDao.
EmployeeService	Реализация интерфейса IEmployeeService.
ITaskService	Интерфейс сервиса, осуществляющий связь класса TaskController с интерфейсом ITaskDao.
TaskService	Реализация интерфейса ITaskService.
IStatisticsService	Интерфейс, содержащий методы для получения статистики по задачам сотрудника или сотрудников.
StatisticsService	Реализация интерфейса IStatisticsService.
RegistrationBean	Представляет собой JavaBeans, используется для упаковки введенных при регистрации данных в один объект для передачи контроллеру.
StatisticsBean	Содержит информацию о статистике задач и соответствующем сотруднике.
Notification	Класс сообщения, объект которого преобразуется в JSON формат для отправки на FCM.
NotificationType	Перечисление типов уведомлений.
RequestSender	Осуществляет отправку запросов, связанных с уведомлениями, на сервер приложений и FCM.

Рассмотрим методы классов, относящихся к системе оповещений. К таким классам относятся TaskController, RequestSender, PushController и Notification. TaskController содержит не только методы, связанные с управлением задач, но и методы для отправки уведомлений. К таким методам относятся:

– private void sendNotificationToPerformer(Task t, NotificationType type) – вызывает метод sendNotification(Task t, NotificationType type, int empid), в который в качестве идентификатора сотрудника передает идентификатор исполнителя задачи; принимает в качестве параметров задачу и тип уведомления;

– private void sendNotificationsToPerformers(Task t, NotificationType type, int cur_emp) – вызывает метод sendNotificationToPerformer(Task t, NotificationType type) для каждого элемента дерева подзадач, если исполнитель задачи не является текущим авторизованным сотрудником;

– private void sendNotificationToCreator(Task t, NotificationType type) – вызывает метод sendNotification(Task t, NotificationType type, int empid), в который в качестве идентификатора сотрудника передает идентификатор создателя задачи; принимает в качестве параметров задачу и тип уведомления;

– private void sendNotification(Task t, NotificationType type, int empid) – в зависимости от типа уведомления формирует его содержимое, данные для уведомления извлекает из переданной задачи; По идентификатору сотрудника, получает список его устройств, формирует сообщения для отправки на FCM; затем вызывает методы для отправки запросов на FCM и на сервер приложений;

– private Notification createNotification(Task t, NotificationType type, List<String> reg_ids) – создает уведомление, с заголовком, текстом и списком устройств; Заголовок и текст извлекаются из файлов ресурсов по типу передаваемого уведомления.

Теперь рассмотрим методы класса RequestSender. Данный класс содержит метод public static List<String> postToFCM(Notification notification),

которые отправляет POST-запрос серверу уведомлений. В качестве сервера уведомлений был выбран Firebase Cloud Messaging (FCM). Передаваемый в метод параметр содержит в себе строковые идентификаторы устройств. Формат данных в запросе – JSON. Метод возвращает список не зарегистрированных строковых идентификаторов устройств, если таковые имеются.

Класс PushController включает в себя следующие методы:

– @GetMapping(value = "/regId") public void addDevice(@RequestParam String endpoint, HttpServletRequest req),

– метод контроллера, обрабатывающий GET-запрос для регистрации идентификатора устройства для пользователя; Запрос, содержит информацию об идентификаторе устройства.

Класс Notification в качестве приватного поля содержит список строковых идентификаторов устройств, а также поле типа Map для сохранения необходимых данных для отображения уведомления (заголовок, описание), а также методы для получения списка и установки значения данного поля.

3.2 Разработка сценариев работы системы оповещения пользователей сети

При разработке приложения было выделено два этапа:

– реализация системы управления задачами;

– реализация системы оповещения.

Для разработки системы управления задачами необходимо было разработать клиентскую и серверную часть. Клиентская часть включает в себя набор веб-страниц, отображаемых пользователю в браузере. Серверная же часть должна осуществлять обработку запросов и взаимодействие с базой данных. Данная система должна позволять добавлять, удалять, изменять и копировать задачи. А также позволять осуществлять просмотр списка задач сотрудника и задач его подчиненных, просмотр статистики по своим задачам

и задачам сотрудников. Помимо перечисленного функционала, приложение должно обеспечивать сортировку задач по различным критериям, просмотр информации о сотруднике, авторизацию и регистрацию сотрудников, поиск по имени задачи и фильтрацию задач по заданному критерию.

На рисунке 15 отображена стартовая страница приложения, она же является страницей авторизации пользователей, т.е. сотрудников.

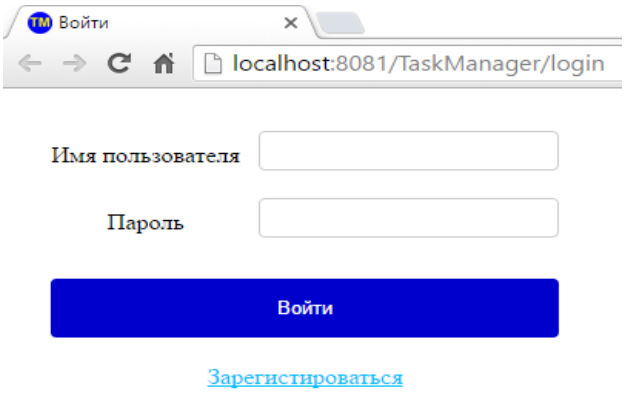

Рисунок 15 - Стартовая страница приложения

Со стартовой страница, являющей страницей авторизации, можно перейти на страницу регистрации, которая показана на рисунке (рисунок 16).

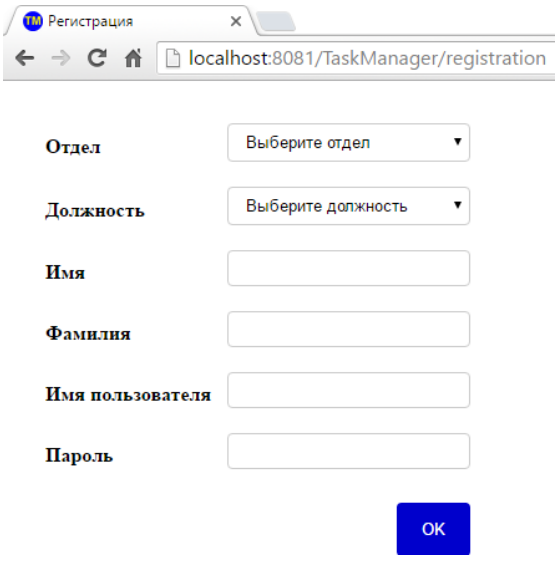

Рисунок 16 - Страница регистрации сотрудника в системе

Из данного рисунка (рисунок 16) видно, что для регистрации необходимо выбрать отдел, затем после выбора будет заполнено поле для выбора должности, который состоит из списка должностей выбранного отдела. Также странице включает в себя поля для ввода имени, фамилии, имени пользователя и пароля, которые не должны быть пустыми. Для передачи данных на сервер необходимо нажать кнопку с надписью «ОК». После успешной регистрации, пользователю будет отображена страница авторизации. Если введенные данные некорректны, то сотруднику отображается соответствующее сообщение. Важно отметить, что в системе могут зарегистрироваться сотрудники, данные о которых уже имеются в базе данных, иначе говоря, только сотрудники, работающие на предприятии. При успешной авторизации сотруднику будет отображена страница навигации. Вид страницы навигации показан на рисунке 17.

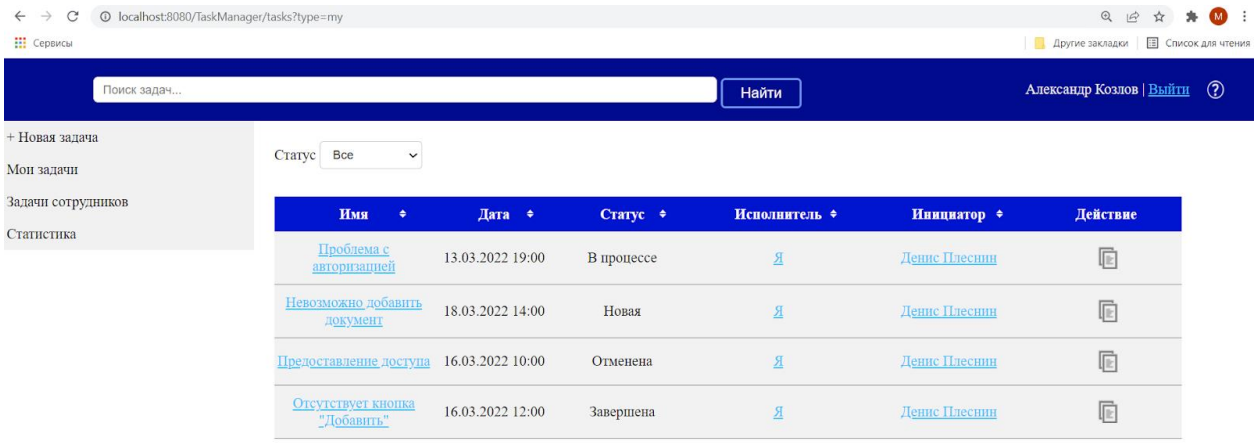

Рисунок 17 - Страница навигации

На данной странице можно осуществить просмотр списка задач авторизованного сотрудника и его подчиненных и статистики по задачам. Также можно осуществить переход на страницы создания задачи, просмотра задачи, просмотра информации о сотруднике. К доступным действиям на странице навигации относятся удаление задачи, копирование, изменение исполнителя из таблицы, сортировка задач, фильтрация задач по статусу и

поиск по имени задачи. Изменение содержимого страницы изменяется динамически в зависимости от выбранного параметра, которым может быть «Мои задачи», «Задачи сотрудников», «Статистика». В таблице со списком задач отображены некоторые данные о задачах. К таким данным относятся идентификатор, имя, дата выполнения, текущий статус и исполнитель. При осуществлении поиска задач отображается эта же страница с таблицей найденных задач. Найденные задачи можно сортировать по критериям, отображенным в таблице, фильтровать по статусу, удалять, копировать и менять исполнителя задачи из таблицы. Копировать можно любую задачу даже такую, для которой авторизованный пользователь не является создателем. При этом статус задач устанавливается как «Новый», а создателем новой задачи является текущий авторизованный сотрудник. Страницу просмотра списка задач подчиненных авторизованного сотрудника можно увидеть на рисунке ниже (рисунок 18).

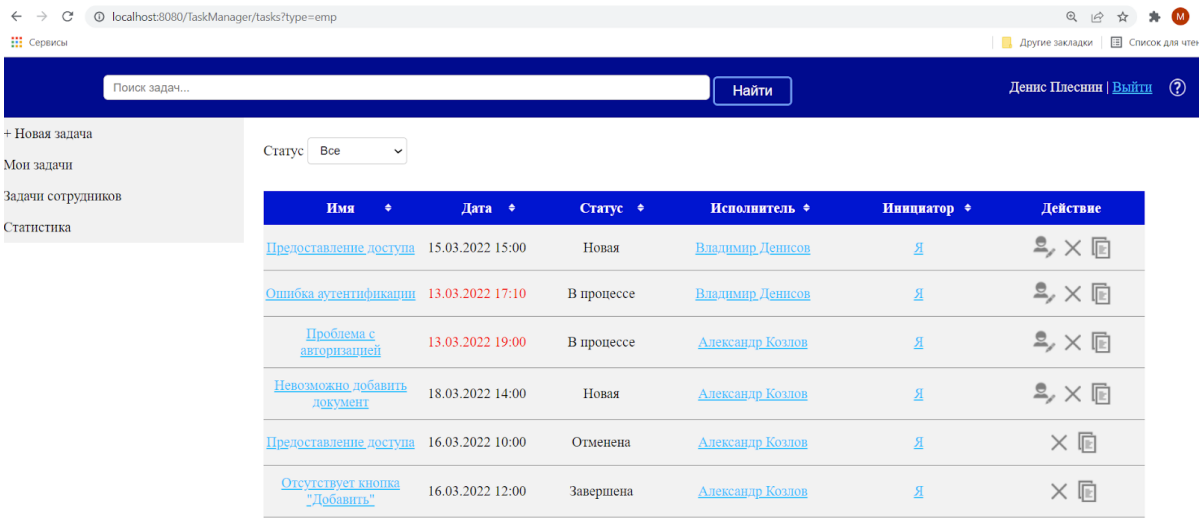

Рисунок 18 - Страница навигации с задачами подчиненных

Выделение даты какой-либо задачи красным цветом означает, что задача является просроченной, т.е. дата уже прошла, а задача еще не завершена. Таблица просмотра статистики по задачам отображена на следующем рисунке (рисунок 19).

Рисунок 19 - Страница навигации со статистикой задач

На данной странице можно посмотреть информацию о количестве текущих, завершенных, отмененных и просроченных и общем количестве задач по авторизованному сотруднику и каждому его подчиненному. Такая статистика может послужить для оценки эффективности работы каждого сотрудника.

Система управления задачами позволяет добавлять задачи. Для добавления задачи была создана соответствующая страница, показанная на рисунке 20.

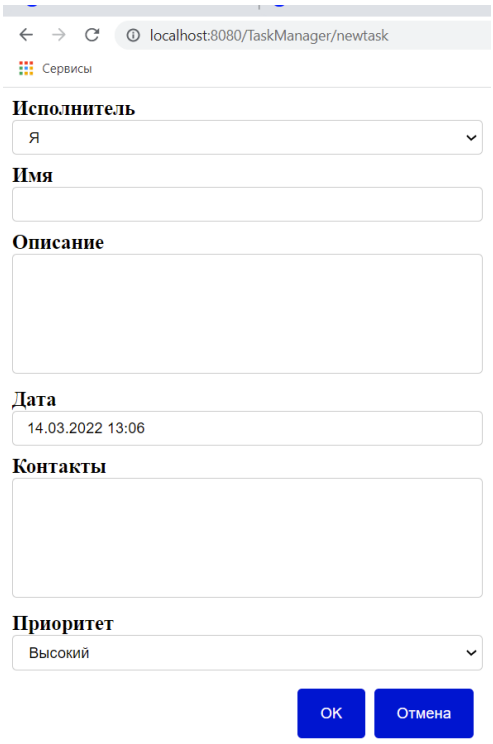

Рисунок 20 - Страница создания задачи

Из рисунка (рисунок 20) видно, что на странице имеются поля для ввода и выбора параметров задачи. Обязательным данными для ввода являются имя и дата выполнения. При корректном вводе параметров созданная задача сохраняется в таблице задач. Страница изменения задачи аналогична странице создания. Однако есть отличия, которые можно увидеть на следующем рисунке (рисунок 21). Если текущий авторизованный сотрудник является и создателем, и исполнителем задачи, то для него доступно изменения всех полей и добавление подзадач. Если же авторизованный пользователь, является только создателем, то он может менять все поля кроме статуса. В случае, когда авторизованный сотрудник является только исполнителем, он может изменять статус задачи и добавлять подзадачи. После отмены задачи имеется возможность ее активировать, после чего задача снова становится актуальной.

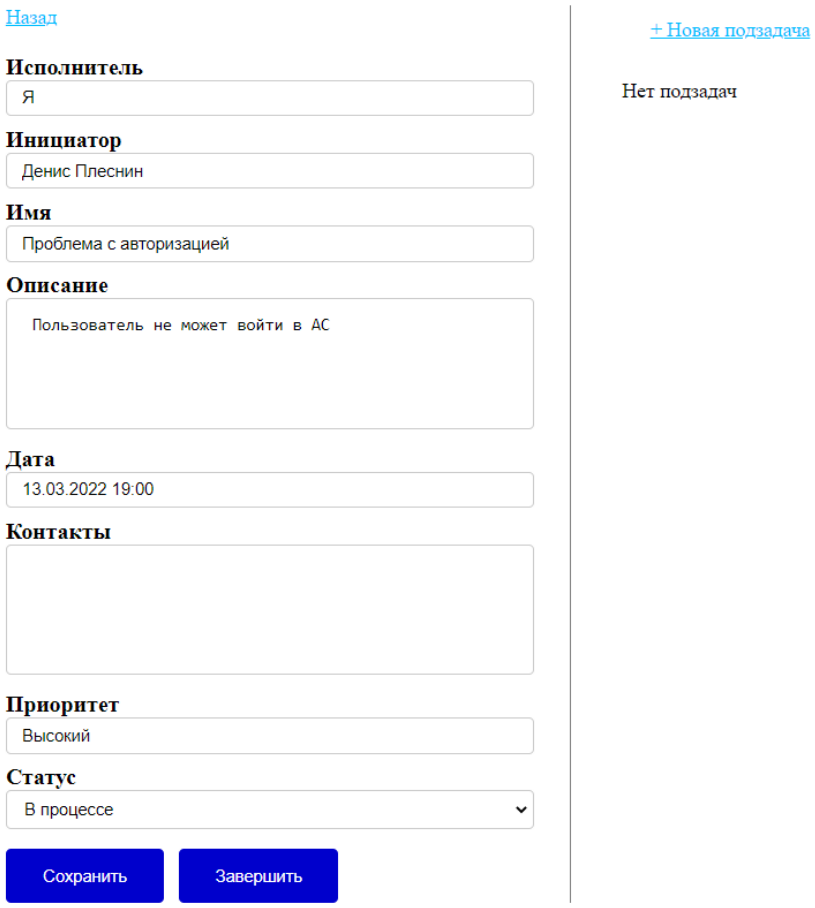

Рисунок 21 - Страница просмотра и изменения задачи

Разработанное приложение позволяет не только просматривать информацию о задаче, но и сотрудниках. Страница просмотра информации о сотруднике показана на рисунке (рисунок 22).

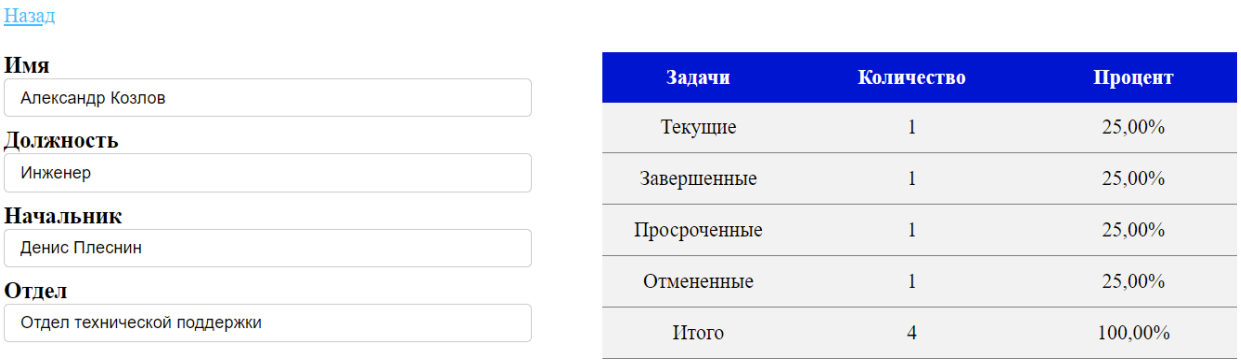

Рисунок 22 - Страница просмотра информации о сотруднике

На данной странице отображены различные параметры сотрудника такие, как имя, должность, имя начальника и отдел, в котором работает сотрудник и статистика по его задачам. Таким образом, были рассмотрены все веб-страницы, которые в совокупности представляют собой клиентскую часть системы управления задачами. Обработку запросов, связанных с системой управления задачами, осуществляют классы EmployeeController и TaskController.

Разработанное приложение также включает систему оповещения сотрудников. Так как системы оповещения представляет собой систему браузерных push-уведомлений, для реализации которой были использованы технологий браузера Chrome, то были выделены следующие этапы разработки:

– создание проекта в консоли разработчика Firebase и подключение к проекту FCM API;

– создание манифеста веб-приложения и подключение к системе управления задачами;

– реализация Service Worker и подключение его к системе управления задачами;

– реализация отправки запроса на сервер приложений и FCM.

Первый этап был реализован после регистрации в консоли разработчика Firebase. После создания проекту ему был присвоен номер. Затем после подключения к созданному ранее проекту FCM API был выдан ключ в виде зашифрованной строки. Для осуществления второго этапа был создан json-файл, который помимо информации о названии и иконки приложения содержал параметр gcm_sender_id. В качестве значения последнего был установлен номер проекта из первого этапа. После создания манифеста веб-приложения он был подключен к странице навигации. Содержание файла манифеста приведено в Приложении А (см. Приложение А). Реализация Service Worker подразумевает создание двух файлов, написанных на JavaScript. Один из этих файлов содержит функционал Service Worker, в который входит получение push-сообщений от FCM, отображение оповещения и обработка нажатия на оповещение. Второй же осуществляет регистрацию файла с функционалом в браузере Chrome. Подключение к системе управления задачами означает подключение к странице навигации файла, который регистрирует Service Worker. Если пользователь заходит в веб-приложение первый раз, и на данном устройстве не было настройки на получение уведомлений, то появляется окно о разрешении или запрете на получение оповещений от веб-приложения. Последний этап, который включает в себя реализацию запросов на сервер приложений и FCM был осуществлен с использованием инструментария Java, позволяющего отправлять POST-запросы. Реализация данного этапа представлена в виде двух методом, содержащихся в классе RequestSender. Листинг класса представлен в Приложении А (см. Приложение А).

Были выделены несколько типов оповещений, которые можно разделить на три группы:

– оповещение исполнителя задачи;

– оповещение создателя задачи;

– оповещение исполнителей подзадач.

К оповещению исполнителя задача относятся следующие выделенные типы:

– оповещение о создании задачи или подзадачи;

– оповещение об изменении даты;

– оповещение об изменении приоритета;

– оповещение об изменении даты и приоритета;

– оповещение об отмене задачи;

– оповещение об активации задачи;

– оповещение о переназначении задачи;

– оповещение об удалении задачи.

При удалении задачи осуществляется также удаление ее подзадач, тогда оповещаются и исполнители подзадач.

К оповещениям создателя задачи относятся:

– оповещение о смене статуса;

– оповещение о завершении задачи.

Оповещение о переназначении осуществляется для предыдущего исполнителя в виде оповещения об отмене (рисунок 25), а для нового исполнителя о создании задачи (рисунок 23). Оповещение об удалении подзадачи аналогично оповещению об удалении задачи (рисунок 24).

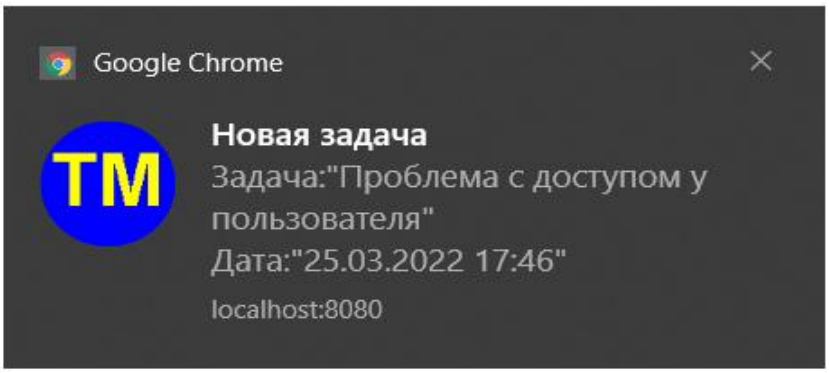

Рисунок 23 - Создание задачи

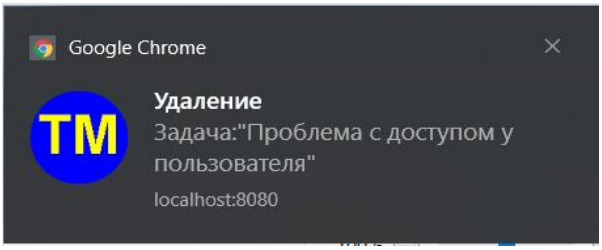

Рисунок 24 - Удаление задачи

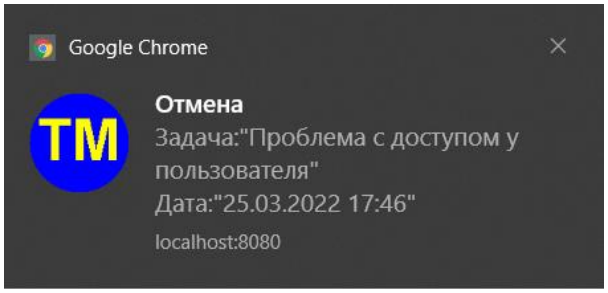

Рисунок 25 - Отмена задачи

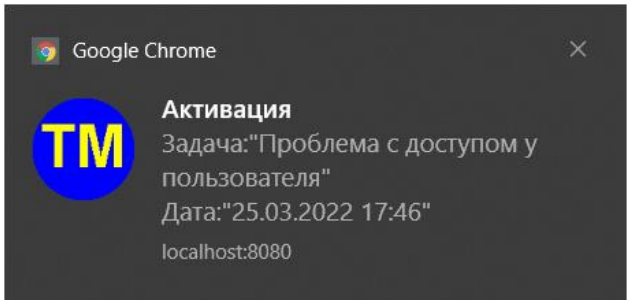

Рисунок 26 - Активация задачи

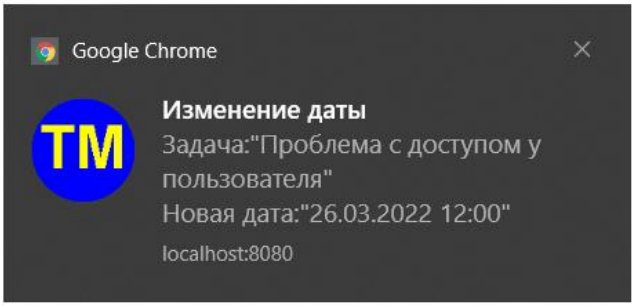

Рисунок 27 - Изменение даты

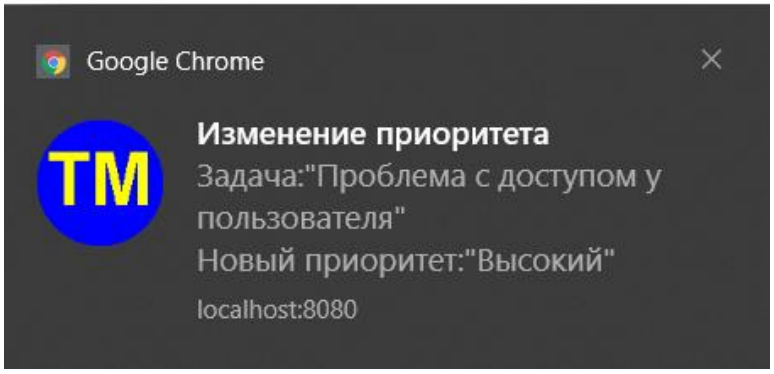

Рисунок 28 - Изменение приоритета

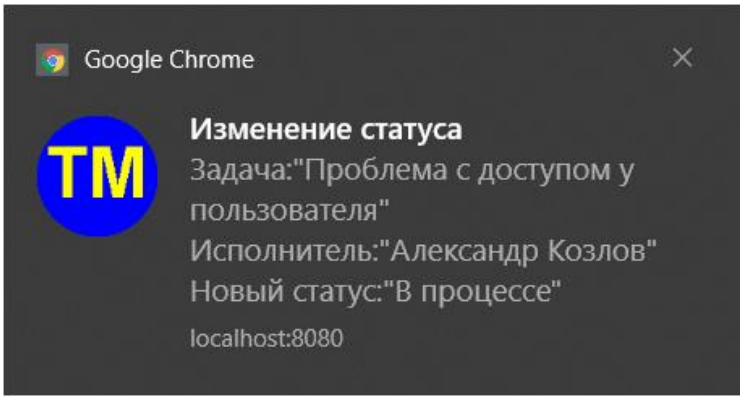

Рисунок 29 - Изменение статуса

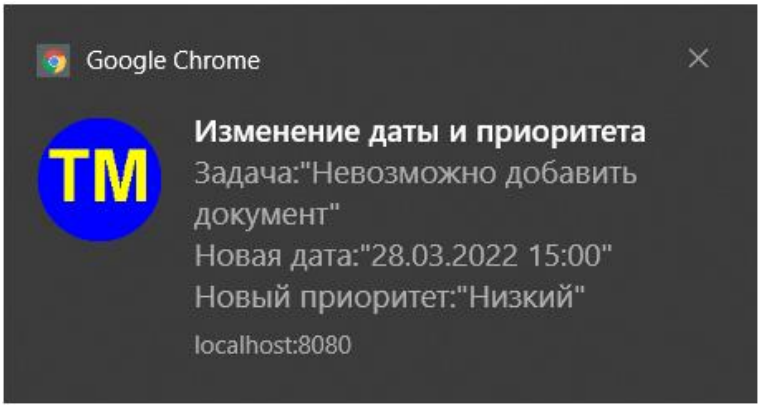

Рисунок 30 - Изменение даты и приоритеты

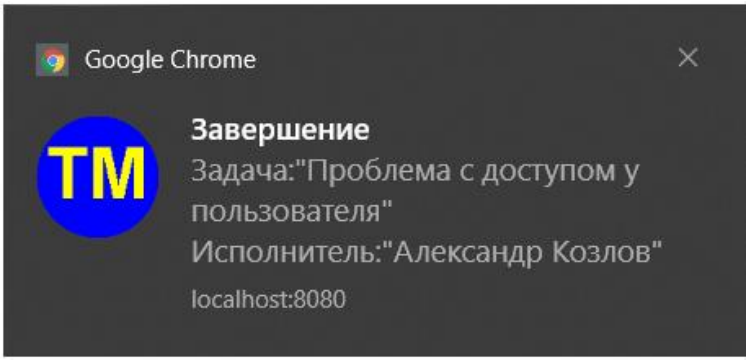

Рисунок 31 - Завершение задачи

Оповещение отображается после того, как получено сообщение с сервера уведомлений. За отображение уведомлений и извлечение информации для него с сервера приложений отвечает Service Worker, который представляет собой файл, написанный на JavaScript. Листинг файлов Service Worker и его регистрации представлены в Приложении (см. Приложение А). Таким образом, были рассмотрены типы реализованных оповещений и соответствующее графическое представление для каждого из них.

3.3 Описание алгоритма работы системы оповещения

При возникновении одной из ситуаций для оповещения, перечисленных в предыдущем параграфе (см. параграф 3.2), вызывается один из методов для формирования и отправки запроса. В методе создаются объект класса Notification. После создания данного объекта, устанавливаются значения его полей. Затем каждый из сформированных объектов преобразуется в строку формата JSON. После чего происходит отправка POST-запроса на FCM, в содержимое, которого входит JSON-строка объекта класса Notification, содержащего идентификаторы устройств пользователя и содержимое push-уведомления, а именно заголовок и описание. Также в заголовке POST-запроса публичный ключ, выданный на пользование FCM. Метод, отправляющий запрос FCM, возвращает список незарегистрированных устройств. Устройства

из данного списка удаляются из базы данных, так как они уже не являются актуальными. После получения запроса FCM отправляет ответ. Затем по извлеченному из запроса идентификатору FCM определяет устройство и отправляет ему push-сообщение. Браузер находит необходимого обработчика push-сообщений, т.е. Service Worker, и отправляет ему. Далее Service Worker извлекает заголовок и текст уведомления и отображает его пользователю. После отображения оповещения пользователь может на него кликнуть, вследствие чего откроется страница авторизации, если пользователь не авторизован, или страница навигации.

3.4 Тестирование приложения

Так как система оповещения, входящая в разработанное приложение, была реализована для браузера Chrome, то тестирование проводилось на устройствах с установленным браузером Chrome, причем версия данного браузера должна быть выше 42. Тестирование проводилось в браузере Chrome версии 102. В ходе тестирования также было проведено исследование, касающееся корректного функционирования системы оповещения в других браузерах. В качестве результатов бакалаврской работы были получены следующие результаты. Система оповещения функционирует корректно в браузерах Edge (версия выше 17), Firefox (версия выше 44), Opera (версия выше 44), Yandex (версия выше 16) [20] [21]. Исправная работа системы зависит от следующих факторов:

– наличие актуальной версии браузера Chrome, Yandex, Edge, Firefox, Opera на устройстве;
– наличие подключения устройства к Интернету;
– постоянно запущенный сервер приложений.

Разработанная система оповещения была протестирована как при выполнении всех условий, так и при отсутствии какого-либо из них. В рамках тестирования были проверены следующие случаи:

– Создание задачи: отправляется оповещение о созданной задаче со следующим содержимым: наименование, контрольный срок созданной задачи и заголовок «Новая задача»;

– Изменение задачи:

a. Изменение приоритета: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, новый приоритет и заголовок «Изменение приоритета»;

b. Изменение контрольного срока: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, новый контрольный срок и заголовок «Изменение даты»;

c. Изменение приоритета и контрольного срока: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, новый приоритет, новый контрольный срок и заголовок «Изменение даты и приоритета»;

d. Изменение статуса: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, новый статус и заголовок «Изменение статуса»;

e. Активация задачи: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, контрольный срок и заголовок «Активация»;

f. Отмена задачи: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи, контрольный срок и заголовок «Отмена»;

g. Завершение задачи: отправляется оповещение об измененной задаче со следующим содержимым: наименование задачи , исполнитель и заголовок «Завершение»;

– Удаление задачи: отправляется оповещение об удаленной задаче со следующими данными: наименование задачи и заголовок «Удаление».

Все тесты, выполнявшиеся при наличии всех трех факторов, были успешно пройдены. Отсутствие какого-либо из перечисленных условий повлечет за собой неработоспособность системы оповещения.

Выводы по главе 3

В третьей главе данной работы была описана реализация системы оповещения пользователей сети.

Описана структура спроектированной базы данных. В базе данных помимо таблиц были созданы последовательности для генерации значений идентификаторов.

Разработанная система состоит из классов, осуществляющих логику работы приложения, а также страниц, отображаемых пользователю, файлов с таблицами стилей страниц, а также файлов, осуществляющих обработку действий пользователя в браузере.

Для системы управления задачами была разработана клиентская и серверная части приложения. Клиентская часть включает в себя набор веб-страниц, отображаемых пользователю в браузере. Серверная часть осуществляет обработку запросов и взаимодействие с базой данных. Приложение позволяет добавлять, удалять, изменять и копировать задачи. А также позволять осуществлять просмотр списка задач сотрудника и задач его подчиненных, просмотр статистики по своим задачам и задачам сотрудников. Для реализации указанных действий на стороне клиентской части были разработаны веб-страницы.

Разработанное приложение включает систему оповещения, реализованную на основе браузерных push-уведомлений, для реализации которой были выделены этапы разработки:

– создание проекта в консоли разработчика Firebase и подключение к проекту FCM API;

– создание манифеста веб-приложения и подключение к системе управления задачами;

– реализация Service Worker и подключение его к системе управления задачами;

– реализация отправки запроса на сервер приложений и FCM.

Были выделены, а затем и реализованы оповещения по следующим группам:

– оповещение исполнителя задачи;

– оповещение создателя задачи;

– оповещение исполнителей подзадач.

Также описаны условия и алгоритмы работы системы оповещения. Тестирование проведено в браузерах, поддерживающих push-уведомления, а именно Google Chrome, Yandex Browser, Firefox, Opera и Microsoft Edge. В данных браузерах приложение для оповещения сотрудников функционирует корректно.

Заключение

В рамках ВКР были проанализированы вопросы, связанные с системами оповещения, выявлены их классификационные признаки, определены способы оповещения пользователей в сети.

Было осуществлено проектирование программного решения в виде системы оповещения и системы управления задачами сотрудников, а также проведен анализ и выбор технологий, которые были использованы для реализации приложения для оповещения и управления задачами сотрудников.

Были представлены модель разработанного приложения, описание структур таблиц базы данных, классов, основной алгоритм работы и графический интерфейс приложения.

Все поставленные цели и задачи были выполнены. Результатом выполнения выпускной квалификационной работы является разработанное приложение для оповещения и управления задачами сотрудников.

Для системы управления задачами была разработана клиентская и серверная части приложения. Клиентская часть включает в себя набор веб-страниц, отображаемых пользователю в браузере. Серверная часть осуществляет обработку запросов и взаимодействие с базой данных. Приложение позволяет добавлять, удалять, изменять и копировать задачи. А также позволять осуществлять просмотр списка задач сотрудника и задач его подчиненных, просмотр статистики по своим задачам и задачам сотрудников. Для реализации указанных действий на стороне клиентской части были разработаны веб-страницы.

Реализованная в рамках приложения система оповещения позволяет получать оповещения о новых задачах, об изменении даты и/или приоритета задачи, о переназначении задачи, об удалении, отмене и активации задачи. Содержимое оповещений соответствует данным о задаче.

Были рассмотрены основные способы оповещения пользователей в сети, исследованы и описаны технологии, позволяющие реализовать веб-

приложение и систему оповещения для него. Описаны условия и алгоритмы работы системы оповещения.

Тестирование разработанного приложения проведено в браузерах, поддерживающих push-уведомления, а именно Google Chrome, Yandex Browser, Firefox, Opera и Microsoft Edge. В данных браузерах приложение для оповещения сотрудников функционирует корректно.

В качестве основных перспектив развития можно выделить следующие направления:

– оптимизация алгоритма на большее количество сотрудников и задач;

– расширение функционала системы управления задачами;

– добавление возможности фильтрации и поиска по разным параметрам задач и сотрудников;

– добавление возможности просмотра статистики по задачам за разные промежутки времени;

– добавление функции, позволяющей осуществлять экспорт статистики по задачам в файл;

– разработка мобильной версии приложения.

Список используемых источников

1. Анатомия новых push-уведомлений Google Chrome и преимущества для сайтов [Электронный ресурс]. URL: https://vc.ru/p/push-chrome (дата обращения: 02.11.2022).
2. Гонсалвес Э. Изучаем Java EE 7. : учеб. пособие для СПО. СПб.: Питер, 2016. — 640с.
3. Использование API уведомлений [Электронный ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API/Using_the_Notifications_API (дата обращения: 12.03.2023).
4. Козмина Ю., Харроп Р., Шефер К., Хо К. Spring 5 для профессионалов. – М.: Диалектика-Вильямс, 2019. — 1120 с.
5. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2019. — 432 с.
6. Советов, Б. Я. Базы данных : учебник для прикладного бакалавриата / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — М. : Издательство Юрайт, 2019. — 420 с.
7. Стасышин, В. М. Базы данных: технологии доступа : учеб. пособие для СПО / В. М. Стасышин, Т. Л. Стасышина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2018. — 164 с.
8. Сысолетин, Е. Г. Разработка интернет-приложений : учеб. пособие для СПО / Е. Г. Сысолетин, С. Д. Ростунцев. — М. : Издательство Юрайт, 2019. — 90 с.
9. Фримен Эрик, Фримен Элизабет, Сиерра К., Бейтс Б. Паттерны проектирования. СПб.: Питер, 2021. — 656 с.
10. Хорстманн К. Java SE 9. Вводный курс 2-х томник. М.: Вильямс, 2020. — 576 с.

11. Эккель Б. Философия Java. 4-е издание. СПб.: "Питер", 2019. — 1168 с.

12. 3 Types of Notifications [Электронный ресурс]. URL: https://bubba.vc/2014/09/03/3-types-of-notifications/ (дата обращения: 10.12.2022).

13. All standarts and drafts (W3C) [Электронный ресурс]. URL: http://www.w3.org/TR (дата обращения: 15.02.2023).

14. Introduction to Service Worker: How to use Service Worker - HTML5 Rocks [Электронный ресурс]. URL: http://www.html5rocks.com/en/tutorials/service-worker/introduction/ (дата обращения: 25.01.2023).

15. Introduction to SSL [Электронный ресурс]. URL: https://www.ibm.com/docs/en/ztpf/2020?topic=started-introduction-ssl (дата обращения: 25.01.2023).

16. Java Technologies for Web Applications [Электронный ресурс]. URL: http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html

17. Maven – Maven documentation [Электронный ресурс]. URL: http://maven.apache.org/guides/ (дата обращения: 01.02.2023).

18. Modelling Notification System in PHP [Электронный ресурс]. URL: http://culttt.com/2015/01/19/modelling-notification-system-php/ (дата обращения: 01.02.2023).

19. Oracle documentation [Электронный ресурс]. URL: https://docs.oracle.com/en/java/index.html (дата обращения: 15.03.2023).

20. Spring Boot documentation [Электронный ресурс]. URL: https://spring.io/projects/spring-boot (дата обращения: 01.04.2023).

21. Spring documentation [Электронный ресурс]. URL: https://spring.io/projects/spring-framework (дата обращения: 01.04.2023).

22. Web Push Notifications: Timely, Relevant, and Precise [Электронный ресурс].

URL: https://developers.google.com/web/fundamentals/push-notifications (дата обращения: 01.03.2023).

23. What Is a Notification System? [Электронный ресурс]. URL: http://www.wisegeek.com/what-is-a-notification-system.htm (дата обращения: 01.12.2022).

24. Spring Boot documentation [Электронный ресурс]. URL: https://spring.io/projects/spring-boot (дата обращения: 02.04.2023).

25. Thymeleaf Documentation [Электронный ресурс]. URL: https://www.thymeleaf.org/documentation.html (дата обращения: 02.04.2023).

Приложение А

Исходный код системы оповещения пользователей сети

Листинг файла Service Worker (sw.js)

```
importScripts('https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/8.10.0/firebase-messaging.js');

console.log('Started', self);

firebase.initializeApp({
 apiKey: 'api-key',
 authDomain: 'project-id.firebaseapp.com',
 databaseURL: 'https://project-id.firebaseio.com',
 projectId: 'project-id',
 storageBucket: 'project-id.appspot.com',
 messagingSenderId: 'sender-id',
 appId: 'app-id',
 measurementId: 'G-measurement-id',
});


const messaging = firebase.messaging();

messaging.onBackgroundMessage((payload) => {
 console.log('Received background message ', payload);
 const title = payload.data.title;
 const message = payload.data.text;
 const data1 = {
 empid: payload.data.empid,
 msg: payload.data.text
 };
 return self.registration.showNotification(title, {
 body: message,
 icon: '/TaskManager/images/icon.png',
 data: data1
 });
});


self.addEventListener('notificationclick', function (event) {

 event.notification.close();
```

```
const url = '/TaskManager/tasks?type=my';
event.waitUntil(
clients.matchAll({
type: 'window'
})
.then(function (windowClients) {
for (let i = 0; i < windowClients.length; i++) {
const client = windowClients[i];
if (client.url.contains('TaskManager')) {
return client.focus();
}
}
if (clients.openWindow) {
return clients.openWindow(url);
}
})
);
});
```

Листинг файла регистрации Service Worker (reg-service-worker.js)

```
if ('serviceWorker' in navigator) {
const firebaseConfig = {
apiKey: "AIzaSyDn6wpQP3-qL_n6zDKMU8ceI_p8TXhPzjw",
authDomain: "task-manager-71d73.firebaseapp.com",
projectId: "task-manager-71d73",
storageBucket: "task-manager-71d73.appspot.com",
messagingSenderId: "1037061012216",
appId: "1:1037061012216:web:598ab81842720e2c5e590d",
measurementId: "G-0R57ZLE5NT"
};


firebase.initializeApp(firebaseConfig);
const messaging = firebase.messaging();

navigator.serviceWorker.register('/TaskManager/js/sw.js').then(function
(registration) {
console.log('Service Worker is registered', registration);
messaging.useServiceWorker(registration);
```

```
 messaging.getToken({
 serviceWorkerRegistration: registration,
 vapidKey: 'BG7aD9ga3q8N0xxjYC-NFBG6-
bZMpctDlmVNUJIPOlR85ZFRqr1tOnqfYCRfh5GqCR_fwrMPdhEJ3XvtwkqbPw
8'
 })
 .then((currentToken) => {
 if (currentToken) {
 console.log( "Current token:" + currentToken);
 $.get("/TaskManager/regId", {'endpoint': currentToken});
 } else {
 console.log('No registration token available. Request permission to generate
one.');
 }
 }).catch((err) => {
 console.log('An error occurred while retrieving token. ', err);
 });

 messaging.onMessage((payload) => {
 console.log('Message received. ', payload);
 const title = payload.data.title;
 const message = payload.data.text;
 const data1 = {
 empid: payload.data.empid,
 msg: payload.data.text
 };
 return registration.showNotification(title, {
 body: message,
 icon: '/TaskManager/images/icon.png',
 data: data1
 });
 })

 }).catch(function (error) {
 console.log('error', error);
 });
}
```

Манифест веб-приложения (manifest.json)

```
{
 "name": "Task manager",
 "icons": [{
 "src": "images/icon.png",
 "sizes": "100x100",
 "type": "image/png"
 }],
 "gcm_sender_id": "1037061012216"
}
```

Листинг класса RequestSender

```
package com.example.taskmgr.utils;


import com.google.gson.*;
import org.springframework.http.*;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

@Component
public class RequestSender {
 public static final String API_KEY =
"AAAA8XWnOvg:APA91bHxPUPpc5051SZIPFggL2gVd7XF4yw_zds26VDnX
0JP4CZmQWtCRVwYQ3-
AXvvjbsoz5m6H6YundWeq7mosT_qRvndl0GYSHbKiLd3ziSi39Hvng61V_9Vjo
FHqlX4pqwLKT5jW";

 public List<String> postToFCM(Notification notification) {
 List<String> not_registered = new ArrayList<>();
 RestTemplate restTemplate = new RestTemplate();
```

```
 HttpHeaders headers = new HttpHeaders();
 headers.setAccept(Collections.singletonList(MediaType.APPLICATION_JSON));
 headers.add("Authorization", "key=" + API_KEY);


 Gson gson = new Gson();
 String s = gson.toJson(notification);

 System.out.println("Request: " + s);

 HttpEntity<Notification> entity = new HttpEntity<>(notification, headers);
 ResponseEntity<String> result =
restTemplate.exchange("https://fcm.googleapis.com/fcm/send",
HttpMethod.POST, entity, String.class);

 int responseCode = result.getStatusCodeValue();
 System.out.println("\nSending 'POST' request to URL :
https://fcm.googleapis.com/fcm/send");
 System.out.println("Response Code : " + responseCode);
 System.out.println("Response: " + result.getBody());

 JsonObject obj = JsonParser.parseString(result.getBody()).getAsJsonObject();

 JsonArray arr = obj.getAsJsonArray("results");

 for (int i = 0; i < arr.size(); i++) {
 JsonObject ob = arr.get(i).getAsJsonObject();
 if (ob.get("error") != null) {
 not_registered.add(notification.getRegistrationIds().get(i));
 }
 }

 return not_registered;
 }
}
```

Скрипт создания таблиц базы данных

```
create table department
(
 deptid bigint not null
 constraint pk_department
 primary key,
 dname varchar(30)
 constraint uc_department_dname
 unique
);


create table employee
(
 id bigint not null
 constraint pk_employee
 primary key,
 username varchar(255)
 constraint emp_user_fk
 references users,
 fname varchar(255),
 lname varchar(255),
 job varchar(255),
 deptid bigint
 constraint fk_employee_on_deptid
 references department,
 mgr_id bigint
 constraint fk_employee_on_mgr
 references employee
);


create table device
(
 id bigint not null
 constraint pk_device
 primary key,
 registration_id varchar(255),
 employee_id bigint
 constraint fk_device_on_employee
 references employee
);
```

```
create table users
(
 username varchar(50) not null
 primary key,
 password varchar(100) not null,
 enabled boolean default true not null
);

create table authorities
(
 username varchar(50) not null
 references users,
 authority varchar(50) not null
);

create table task
(
 id bigint not null
 constraint pk_task
 primary key,
 name varchar(255),
 tdesc varchar(255),
 tdate timestamp,
 crdate timestamp default now() not null,
 contacts varchar(255),
 state varchar(255),
 creator_id bigint
 constraint fk_task_on_creator
 references employee,
 executor_id bigint
 constraint fk_task_on_executor
 references employee,
 parent_task_id bigint
 constraint fk_task_on_parent_task
 references task,
 priority varchar(255)
);
```