Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчет по преддипломной практике

Студент А.О. Игнатович

Руководитель И.Л. Селезнев

Консультанты:

от кафедры ЭВМ И.Л. Селезнев

Нормоконтролер Е.Е. Клинцевич

МИНСК 2023

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. каф. ЭВМ
_____ Б.В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему
ЛИЧНЫЙ КАБИНЕТ СОТРУДНИКА БГУИР

БГУИР ДП 1–40 02 01 01 307 ПЗ

Студент А.О. Игнатович

Руководитель И.Л. Селезнев

Консультанты:

от кафедры ЭВМ И.Л. Селезнев

по экономической части Т.А Рыковская

Нормоконтролер Е.Е. Клинцевич

Рецензент

МИНСК 2023

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: КСиС. Кафедра: ЭВМ.
Специальность: 40 02 01 «Вычислительные машины, системы и сети».
Специализация: 40 02 01-01 «Проектирование и применение локальных компьютерных сетей».

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Б.В.Никульшин
«____» _____ 2023 г.

ЗАДАНИЕ
по дипломному проекту студента
Игнатович Анны Олеговны

1 Тема проекта: «Личный кабинет сотрудника БГУИР» – утверждена приказом по университету от 3 апреля 2023 г. № 814-с.

2 Срок сдачи студентом законченного проекта: 1 июня 2023 г.

3 Исходные данные к проекту:
3.1 Протокол взаимодействия: HTTP, HTTPS.
3.2 Формат обмена данными: JSON.
3.3 Операционная система: Windows 10 Pro.
3.4 Среда разработки: IntelliJ IDEA.
3.5 Языки программирования: Java, React.

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов):
Введение 1. Обзор литературы. 2. Системное проектирование. 3. Функциональное проектирование. 4. Разработка программных модулей. 5. Программа и методика испытаний. 6. Руководство пользователя. 7. Технико-экономическое обоснование разработки. Заключение. Список использованных источников. Приложения.

5 Перечень графического материала (с точным указанием обязательных чертежей):
5.1 Вводный плакат. Плакат.

5.2 Личный кабинет сотрудника БГУИР. Схема структурная.
5.3 Личный кабинет сотрудника БГУИР. Диаграмма классов.
5.4 Личный кабинет сотрудника БГУИР. Диаграмма последовательности.
5.5 Личный кабинет сотрудника БГУИР. Схема программы.
5.6 Заключительный плакат. Плакат.

6 Содержание задания по экономической части: «Экономическое обоснование разработки и реализации программного модуля личного кабинета сотрудника БГУИР».

ЗАДАНИЕ ВЫДАЛ Т.А Рыковская

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов дипломного проекта	Объем этапа, %	Срок выполнения этапа	Примечания
Подбор и изучение литературы. Сравнение аналогов. Уточнение задания на ДП	10	23.03 – 30.03	
Структурное проектирование	15	30.03 – 08.04	
Функциональное проектирование	25	08.04 – 24.04	
Разработка программных модулей	20	24.04 – 08.05	
Программа и методика испытаний	10	08.05 – 15.05	
Расчет экономической эффективности	5	15.04 – 20.05	
Оформление пояснительной записки	15	20.05 – 30.05	

Дата выдачи задания: 23.03.2023

Руководитель И.Л. Селезнев

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время использование информационных технологий в управлении человеческими ресурсами является одним из ключевых трендов в развитии современных организаций. Одним из важных инструментов в этой сфере являются личные кабинеты сотрудников, которые предоставляют возможность эффективного управления персоналом, повышения производительности и удовлетворенности сотрудников.

Личные кабинеты для сотрудников также широко используются в сфере банковской и финансовой деятельности, где они позволяют сотрудникам управлять своими финансовыми данными, зарплатой, налогами и другими аспектами своей работы.

Данный инструмент является максимально полезным приобретением в любого размера компании, по причине максимальной оптимизации рабочего процесса в разных сферах деятельности работников, а главное – налаживанию коммуникаций между разными отделами, что также положительно и качественно влияет на рабочий процесс.

Цель данного дипломного проекта заключается в разработке личного кабинета сотрудника для Белорусского государственного университета информатики и радиоэлектроники (БГУИР). Данный проект предусматривает создание удобной и функциональной системы, которая позволит сотрудникам университета получать доступ к различного рода информации, касаемой рабочего процесса, поможет оптимизировать коммуникации не только между собой, но также и между отделами, чтобы повысить эффективность и качество взаимодействия. Система будет предусматривать наличие разных прав доступа, чтобы обезопасить данные незапланированных изменений.

В работе рассмотрены основные требования к личному кабинету сотрудника и проведен анализ существующих решений в данной области. Была рассмотрена внутренняя структура взаимодействия сотрудников университета, отделов и принципы их работы для наилучшего понимания и разработки необходимого функционала кабинета, а также закладывание фундамента для последующего расширения в более серьёзную и многофункциональную систему.

В соответствии с целью, поставленной при формулировании концепта дипломного проекта, был определен ряд следующих задач, которые были рассмотрены в разных разделах пояснительной записки:

1. Выбор и обоснование подходящих средств для разработки системы.
2. Разработка логики взаимодействия клиентской части с серверной, а также формирование подходящей им базы данных.
3. Разработка интуитивно понятного пользовательского интерфейса.
4. Тестирование разработанной системы.
5. Создание руководства пользователя.
6. Обоснование экономических затрат на создание данного проекта.

1 ОБЗОР ЛИТЕРАТУРЫ

В данном разделе будут проведены исследования предметных областей, которые затрагиваются в разрабатываемом проекте. Под предметными областями подразумевается используемые методы для создания, а также инструменты и подходы к проектированию.

1.1 Обзор аналогов

Прежде чем рассматривать используемые методы и технологии, которые будут применяться в дипломном проекте, в данном подразделе рассмотрим и проведем анализ существующих аналогов, чтобы разработать план действий и облегчить поиск необходимых материалов. Также это необходимо, чтобы избежать ошибок в проектировании и найти более оптимальные решения.

1.1.1 Sage HR

Sage HR [1] (см. рисунок 1.1) – это компания по разработке программного обеспечения (ПО) для управления персоналом. Оно предназначено как для малых, так и для средний компаний. Управление происходит через веб-сайт, а также мобильное приложение. Каждый пользователь имеет доступ к своему расписанию.

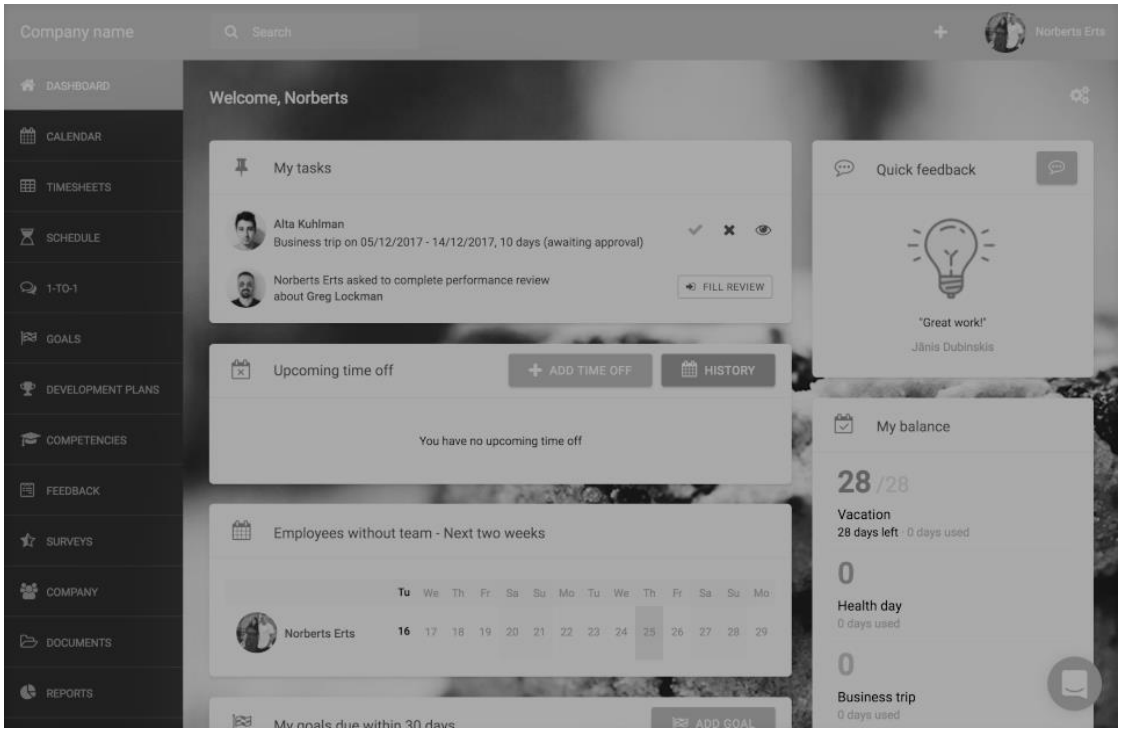

Рисунок 1.1 – Главная страница Sage HR [1]

На главной странице приложения представлены основные функциональные элементы, необходимые для работы пользователя. Дизайн

интерфейса привлекателен и выдержан в ярких тонах, однако, избыточное использование насыщенных цветов может вызвать утомление глаз при продолжительном использовании приложения.

Отсутствие перевода на русский язык может негативно сказаться на работе персонала, так как в университете много людей, которые предпочитают использовать русский язык.

При подключении данного приложения есть возможность подключать только необходимые функции, это значит, что приложение модульное и каждый из модулей работает независимо от другого. Это удобно, потому что не весь функционал необходим, а также будет возможность добавить при необходимости.

В данном приложении сотрудник должен самостоятельно регистрироваться, что может создать некоторые сложности, если возникнут ошибки при заполнении. В случае дипломного проекта это предусмотрено и важную информацию будет изменять либо отдел кадров, то есть человек с привилегиями администратора. Также будет выслан временным пароль, при создании аккаунта, который будет необходимо сменить в течение некоторого времени.

1.1.2 WebHR

WebHR [2] (см. рисунок 1.2) – это онлайн-инструмент для управления персоналом, который охватывает все аспекты работы отдела кадров, от приема на работу до выхода на пенсию. WebHR упрощает HR-процессы за счет шаблонизации, автоматизации и интеграции HR-процессов, чтобы к ним можно было получить доступ и обработать их с единой панели управления.

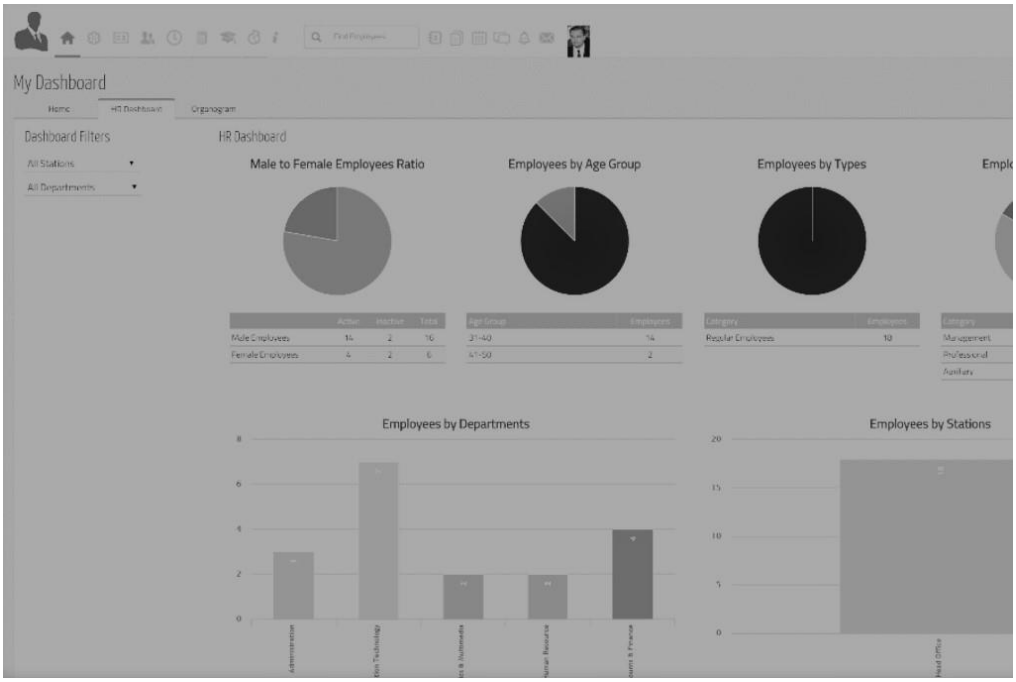

Рисунок 1.2 – Страница WebHR [2]

Одной из возможностей WebHR является интеграция с сайтом поиска работы Indeed.com, которая позволяет автоматически загружать объявления о вакансиях на сайт. Это является удобным и привлекательным решением, но наш регион имеет свои собственные сайты для поиска персонала, что делает эту функцию менее привлекательной в сравнении с данным проектом. Однако стоит принять во внимание данную возможность использования функции в дальнейшем развитии веб-приложения.

За расчет заработной платы отвечает внешний подрядчик, однако все функции для расчета заработной платы присутствуют в приложении. Тем не менее, следует отметить, что приложение ориентировано преимущественно на работу с отделом кадров, что может стать недостатком в контексте необходимости обеспечения эффективного обмена информацией между разными подразделениями и, соответственно, самом использовании разными отделами. Цель данного проекта как раз и заключается в объединении вообще всего персонала в единое целое в силу того, что учебное заведение разделено по разным корпусам, которые расположены далеко друг от друга.

Также следует отметить, что наличие только англоязычной версии является недостатком приложения. Внедрение русской локализации в приложении может значительно улучшить его удобство и доступность пользователей.

С точки зрения визуального оформления, это приложение выглядит более привлекательно, чем первый вариант, благодаря использованию простых и неярких цветов, которые не отвлекают внимание и позволяют ясно видеть все необходимые разделы.

Один из недостатков данного приложения заключается в ограниченной поддержке, что является серьезным недостатком в связи с большим количеством сотрудников в составе БГУИР. В случае возникновения сбоев в работе приложения, это может стать источником значительных неудобств для пользователей.

1.2 Обзор технологий

Данный диплом предполагает разработку системы, которая является веб-приложением, это значит, что это ПО запускается в веб-браузере и оно имеет клиент-серверную архитектуру. Логика такого приложения разделена между сервером и клиентом, а данные, как правило, хранятся на сервере. Обмен информацией между сервером и клиентом происходит через сеть Интернет. Одним из основных преимуществ такого подхода является возможность запуска приложения на разных операционных системах, поскольку клиенты не зависят от конкретной ОС пользователя. В результате веб-приложения могут быть использованы на разных платформах, что делает их универсальными [3].

Система является небольшой и создается одним человеком, поэтому она имеет монолитную архитектуру. Это означает, что различные компоненты, а

именно бизнес-логика, слой доступа к данным, он же подключение к базе данных, интерфейс пользователя и так далее, находятся внутри одного процесса. Даная архитектура проста в развертывании, масштабировании и, соответственно, тестировании.

Если говорить о паттернах проектирования, то в данном веб-приложении использовался паттерн MVC – Model, View, Controller. Приложение будет разделено на три условные части: модель, то есть данные, которые будут передаваться между представлениями и контроллерами; представления, которые будут визуализировать данные модели для пользовательского интерфейса; контроллеры, которые будут обрабатывать запросы и выбирать соответствующие им представления для визуализации [6].

Язык программирования для серверной части – Java, фреймворк Spring. Этот фреймворк содержит много разных модулей

Для клиентской части – React. База данных, для содержания необходимых данных – PostgreSQL. Обо всем этом будет подробнее рассмотрено в отдельных пунктах ниже.

1.2.1 Клиент-серверная архитектура

Клиент-серверная архитектура (см. рисунок 1.3) – это модель взаимодействия между компьютерами в сети, при которой один компьютер (сервер) предоставляет определенные ресурсы или услуги, а другие компьютеры (клиенты) запрашивают эти ресурсы или услуги через сеть.

В клиент-серверной архитектуре обычно выделяют два типа компонентов: клиентские и серверные.

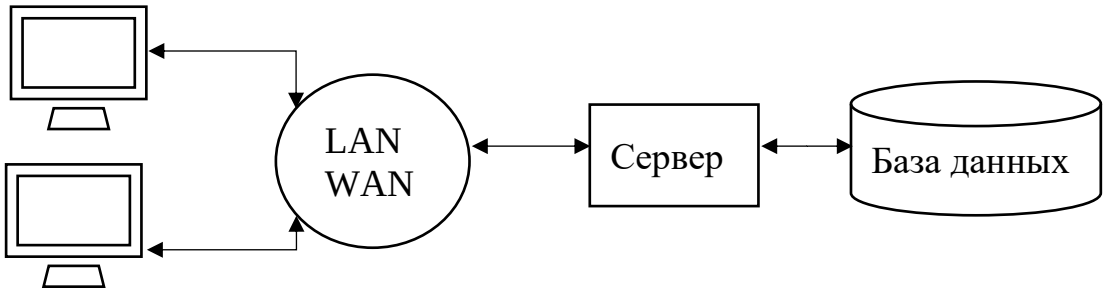

Рисунок 1.3 – Клиент-серверная архитектура

Клиенты – это устройства, на которых запускается пользовательский интерфейс, которые запрашивают ресурсы или функции у серверов. Это конечный пользователь, который использует приложение, и обычно оно работает на его компьютере или мобильном устройстве.

Серверы – это высокопроизводительные вычислительные устройства, на которых размещаются ресурсы и функции, которые клиенты запрашивают.

Серверы могут выполнять различные задачи, например, обрабатывать запросы клиентов, хранить и обрабатывать данные, осуществлять авторизацию и аутентификацию пользователей и так далее.

Взаимодействие между клиентами и серверами происходит через сеть, обычно с использованием протоколов передачи данных, таких как HTTP, FTP, SMTP и тому подобные. Клиентские компоненты отправляют запросы на сервер, и серверные компоненты отвечают на эти запросы, предоставляя запрошенные данные или услуги.

Клиент-серверная архитектура является широко распространенной и используется во многих областях, таких как веб-приложения, базы данных, игровые системы и т.д. Она позволяет создавать масштабируемые и гибкие системы, которые могут обрабатывать большие объемы запросов и предоставлять доступ к данным и услугам из любой точки сети.

Также в разрабатываемой системе присутствует еще один компонент – база данных, программа, в которой хранятся все данные приложения. Этот момент делает данную архитектуру трехуровневой, потому что она состоит из трех компонентов.

В клиент-серверной архитектуре сервер играет роль не только компьютера, на котором работает приложение или сайт, но также является хранилищем всех данных. Клиенты не имеют прямого доступа к базе данных, что защищает их личную информацию и обеспечивает приватность других пользователей, например, в социальных сетях.

Клиенты обращаются к серверу за информацией, и если сервер считает, что пользователь имеет права на получение этой информации, то он ее предоставляет. Это гарантирует защиту личных данных других пользователей [4].

Трехуровневая архитектура информационных систем может быть улучшена путем добавления дополнительных серверов и преобразована в многоуровневую архитектуру. Такая виртуальная архитектура позволяет значительно увеличить эффективность работы информационных систем и оптимизировать использование программно-аппаратных ресурсов [5].

1.2.2 MVC

Шаблон MVC (см. рисунок 1.4) [6] – это шаблон проектирования веб-приложений, который включает в себя три отдельных компонента: модель данных приложения, пользовательский интерфейс и логику взаимодействия пользователя с системой. Это позволяет минимизировать воздействие на другие компоненты при модификации одного из них. Кроме того, MVC включает несколько мелких шаблонов, что делает его более гибким и удобным для использования.

Основной целью использования шаблона проектирования MVC является разделение бизнес-логики и данных от визуализации. Это упрощает повторное использование кода и облегчает сопровождение. Например, если

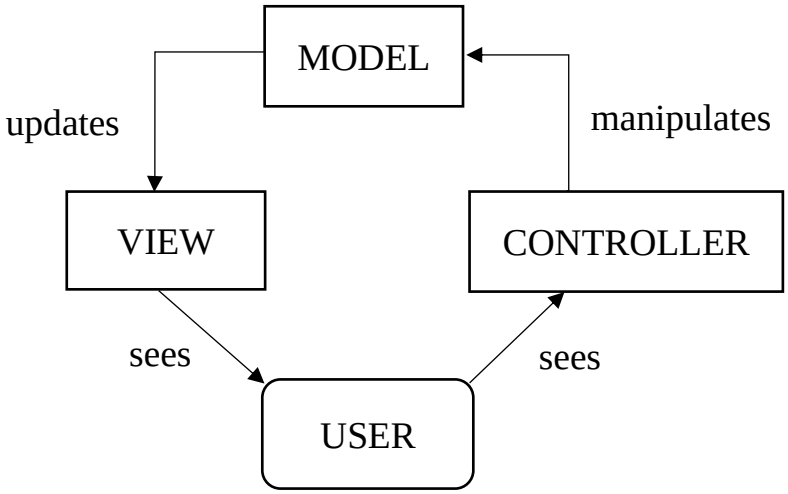

Рисунок 1.4 – Шаблон MVC [6]

необходимо добавить новое представление данных, такое как JSON, XML, PDF или XLSX, к уже существующему маршруту, это можно сделать без изменений в бизнес-логике маршрута. А изменения визуализации не затрагивают бизнес-логику, а изменения бизнес-логики не влияют на визуализацию.

Изменение каждого из модулей происходит независимо. Это означает, что они имеют разные обязанности и ответственности и все поведение будет направлено на выполнение одной задачи. Подобный подход соответствует SOLID-принципу единой ответственности [7].

1.2.3 База данных PostgreSQL

Учитывая особенности проектируемой системы, было необходимо использовать место для хранения данных. Для проекта была выбрана система управления базами данных PostrgreSQL.

PostgreSQL – это популярная свободная объектно-реляционная система управления базами данных (СУБД). Она базируется на языке SQL и поддерживает многочисленные возможности [8].

Преимущества данной СУБД:

- поддержка баз данных неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость.

СУБД обеспечивает высокую надежность и эффективность работы, включая поддержку транзакций (ACID) и встроенные механизмы репликации. Кроме того, система расширяема, что позволяет создавать собственные типы данных и индексы, а также расширять ее функциональность с помощью языков программирования.

Строительство системы на основе PostgreSQL обеспечивает более гладкую интеграцию с другими внутренними системами БГУИР, что является преимуществом для разработки данной системы.

1.2.4 Spring Framework Java

Spring Framework [9]– это один из самых популярных фреймворков для разработки приложений на языке Java, он универсальный, с открытым кодом. Предоставляет комплексный набор инструментов и библиотек для упрощения создания сложных приложений. В какой-то степени его можно назвать фреймворком фреймворков, потому что он состоит из отдельный модулей. В данном веб-приложении использовались такие модули как Spring Data, Spring Security, Hibernate, Spring MVC, JPA. Также Spring позволяет создавать RESRful веб-сервисы. Все эти модули будут рассмотрены подробнее ниже.

REST (от англ. Representational State Transfer — «передача состояния представления») – это общие принципы организации взаимодействия приложения/сайта с сервером посредством протокола HTTP. По сути, взаимодействие с сервером происходит в четыре операции: 1) получение данных с сервера, 2) добавление на сервер новых данных, 3) изменение уже существующих на сервере данных и 4) удаление данных.

Основные принципы такой организации являются:

1. Ресурсы (resources), каждый из которых должен обладать своим уникальным идентификатором URI (Uniform Resource Identifier), который клиент может использовать для доступа к этому ресурсу. Они могут представлять собой данные, которые могут быть получены или как-то изменены с помощью HTTP-методов.

2. Представления (Representation), когда каждый ресурс может иметь несколько представлений, которые определяют формат и содержание данных, возвращаемых сервером в ответ на запрос клиента. Оно может быть в таких форматах как HTML, XML, JSON или любом другом формате, который может быть прочитан клиентом.

3. Методы HTTP (HTTP Methods), для работы с ресурсами, доступны четыре основных метода HTTP: GET для получения данных, POST для создания новых ресурсов, PUT для изменения уже существующих ресурсов и DELETE для удаления ресурсов.

4. Без состояния (Stateless), каждый запрос, отправленный клиентом на сервер, содержит все необходимые данные для его обработки. Сервер не запоминает информацию о предыдущих запросах клиента, что делает реализацию и масштабирование сервера более простым.

Spring Data – это набор инструментов и библиотек, предоставляющих единую модель программирования для доступа к данным. Он использует Spring-подход для упрощения работы с различными типами баз данных, включая реляционные, нереляционные и облачные базы данных.

Основной концепцией является репозиторий, который представляет собой набор интерфейсов для взаимодействия с JPA Entity. Например, интерфейс `CrudRepository<T, ID extends Serializable>`, который расширяет `Repository<T, ID>`, обеспечивает базовую функциональность для создания, чтения, обновления и удаления данных, иначе говоря, обеспечивают CRUD (create, read, update, delete).

Есть абстракции типа `PagingAndSortingRepository`, которая предоставляет методы для разбиения на страницы и сортировки записей. И еще одна `JpaRepository` предоставляет некоторые связанные с JPA методы, такие как очистка контекста постоянства и удаление записей в пакете.

Spring Security – это фреймворк безопасности, который предназначен для использования в приложениях, построенных на платформе Spring. Он предоставляет мощные функции аутентификации и авторизации, которые можно использовать для обеспечения безопасности веб-приложений, микросервисов, REST-сервисов и других приложений на основе Spring.

Основным компонентом Spring Security является фильтр безопасности, который работает на уровне HTTP запросов и ответов. Фильтр может быть настроен для выполнения различных задач, таких как проверка учетных данных пользователя, проверка разрешений доступа к ресурсам, фильтрация запросов и другое. Фильтр обеспечивает безопасность, путем применения цепочки фильтров к каждому запросу, что позволяет выполнять настройку защиты для каждого запроса.

Hibernate – это фреймворк для объектно-реляционного отображения (ORM) в Java, который упрощает доступ к базам данных, позволяя разработчикам работать с объектами, а не с SQL-запросами. Он позволяет сопоставлять объекты Java с таблицами в базе данных и автоматически генерировать соответствующий SQL-код. Он также обеспечивает механизмы для управления отношениями между объектами, транзакциями, кэшированием данных и другими функциями, необходимыми при работе с базами данных.

Преимущества Hibernate включают улучшенную производительность, упрощенное программирование и снижение количества ошибок, связанных с неправильным использованием SQL-запросов. Он также обеспечивает переносимость кода между различными СУБД и улучшенную безопасность при работе с базами данных.

Java Persistence API (JPA) – это спецификация Java EE для управления объектно-реляционным отображением (ORM) в приложениях Java. Она предоставляет стандартный способ описания сущностей, которые могут быть сохранены в базе данных, и управления их жизненным циклом в рамках приложения. JPA облегчает работу с базами данных и ORM-фреймворками, такими как Hibernate, EclipseLink, OpenJPA и другими.

Она определяет аннотации, которые можно добавлять к классам и полям, чтобы указать отображение на реляционную базу данных. Эти

аннотации включают аннотации, такие как `@Entity`, `@Table`, `@Column` и `@Id`, которые используются для описания сущностей и их свойств.

JPA также определяет набор методов для управления жизненным циклом объектов, таких как `persist()`, `merge()`, `remove()`, `refresh()` и `find()`. Он также поддерживает JPQL (Java Persistence Query Language), что позволяет создавать запросы к базе данных, используя объектную модель данных вместо SQL.

Обеспечивает абстракцию от конкретной базы данных, что позволяет легко переносить приложения между различными СУБД без изменения кода приложения. Это делает JPA мощным инструментом для разработки приложений, которые должны работать с различными базами данных и управлять большим объемом данных.

Spring MVC – это фреймворк для разработки веб-приложений на языке программирования Java, основанным на паттерне проектирования Model-View-Controller (MVC), описанном ранее в подразделе выше.

Также он обеспечивает обработку ошибок, валидацию данных, аутентификацию и авторизацию, обработку AJAX-запросов и множество других функций, что делает его одним из наиболее популярных фреймворков для разработки веб-приложений на Java.

1.2.5 React

React [10] является JavaScript-библиотекой для создания пользовательских интерфейсов (UI). Он позволяет создавать компоненты, которые отвечают за отображение данных на странице. Компоненты могут быть многоразовыми и взаимодействовать между собой.

React использует Virtual DOM (Document Object Model или виртуальное дерево объектов), что позволяет обновлять только те элементы, которые действительно изменились, а не обновлять всю страницу целиком. Это повышает производительность и скорость работы приложения.

DOM – это объектная модель документа, которая представляет собой иерархическую структуру документа в виде дерева объектов. DOM дерево состоит из узлов (Node), которые могут быть элементами (Element), атрибутами (Attribute), текстом (Text), комментариями (Comment) и др. Узлы связаны друг с другом иерархически в родительские (parentNode) и дочерние (childNodes) отношения, а также соседние узлы (previousSibling и nextSibling). Позволяет программно создавать, изменять и удалять элементы и их атрибуты, а также реагировать на события, такие как щелчки мыши, изменения размеров и другие. DOM API может быть использован в JavaScript, чтобы создавать интерактивные и динамические веб-страницы.

Это не браузерная технология, это стандарт, определяемый W3C. Браузеры предоставляют DOM API, который можно использовать для манипулирования содержимым документа. DOM API позволяет получать

доступ к элементам документа, изменять их содержимое, атрибуты и стили, добавлять и удалять элементы, а также реагировать на события.

Одним из преимуществ React является большое количество готовых компонентов и библиотек, которые можно использовать в своих проектах. React также хорошо подходит для создания больших приложений, которые могут быть разбиты на множество многоразовых компонентов.

1.3 Постановка задачи

Исходя из анализа аналогов, можно сделать вывод, что для современных программ важна простота, незагруженный и интуитивно понятный интерфейс. Программа должна быть масштабируемая, переносимая, а также мультиплатформенная, чтобы ее можно было запускать на разных системах. Также следует отметить, что должен быть соответствующий нуждам пользователей функционал.

Применение указанных технологий и подходов поможет создать качественную систему и ускорить разработку дипломного проекта, а также обеспечит выполнение требований к проекту.

Следует добавить, что данный дипломный проект создается для учреждения, в котором уже существуют свои внутренние ресурсы и данная система должна не только соответствовать требованиям современного приложения, а также иметь возможность легко интегрироваться в имеющуюся среду. Соответственно, некоторыми из пунктов выполнения данного требования являлось использования СУБД PostgreSQL и фреймворка Spring.

Основные принципы, лежащие в основе данного программного комплекса – это интуитивно понятный и легкий в использовании интерфейс, расширяемый функционал, а также охват нужд сотрудников и облегчение их коммуникаций.

Для дипломного проекта были определены следующие задачи:

- разработка системы таблиц в базе данных и их взаимодействие;
- разработка бизнес-логики для серверной части;
- разработка и реализация пользовательского интерфейса.

Данный программный комплекс будет реализован в виде веб-сайта, доступного через браузер, и предоставлять следующий набор функций:

- регистрация и авторизация пользователя;
- наличие личной страницы сотрудника;
- наличие двух уровней доступа – user и admin;
- просмотр всех сотрудников;
- администрирование аккаунтов;
- заполнение и хранение документов;
- создание событий, а также назначение их на других пользователей;
- создание заданий или просьб для сотрудников;
- возможность отслеживать присутствие сотрудников на работе;
- система фильтрации списка пользователей для удобного поиска.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Исходя из анализа теоретической части разрабатываемой системы был выделен ряд требований, которые необходимо выполнить для обеспечения стабильного и эффективного функционирования системы. Получив данный список, было принято решение разделить систему на функциональные блоки. Данный подход удобен тем, что внесение изменений в один блок не требует изменения системы в целом. Иными словами, изменяя один блок, остальные остаются нетронутыми и не нуждаются в правках. Это значительно сэкономит время, а также позволит упростить и сделать сам процесс разработки удобнее.

Последующие блоки были выделены в данном дипломном проекте:

- блок базы данных;
- блок взаимодействия с базой данных;
- блок пользовательского интерфейса;
- блок взаимодействия пользовательского интерфейса с контроллерами;
- блок контроллеров;
- блок бизнес-логики;
- блок авторизации пользователей;
- блок user;
- блок admin.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.307 С1.

Ниже рассматриваются вышеперечисленные блоки веб-приложения более подробно.

2.1 Блок базы данных

Стоит упомянуть, что СУБД PostgreSQL – это реляционная база данных, которая обладает множеством преимуществ перед другими СУБД. Она поддерживает ACID-принципы (Atomicity, Consistency, Isolation, Durability), что обеспечивает надежность хранения данных. Также PostgreSQL обладает высокой производительностью благодаря использованию индексов и интеллектуальному планировщику запросов, а также расширяемости, которая позволяет пользователю определять новые функции и типы данных. Кроме того, PostgreSQL поддерживает язык SQL, а также JSON и имеет богатый набор типов данных. В целом, PostgreSQL является простой в использовании и гибкой СУБД, которая подходит для широкого спектра задач.

Реляционная база данных (РБД) – это база данных, основанная на реляционной модели данных. РБД состоит из таблиц (реляций), которые связаны между собой ключами. Каждая таблица в РБД представляет собой двумерную структуру, которая состоит из строк (кортежей) и столбцов (атрибутов). Каждая строка представляет собой набор значений атрибутов, а каждый столбец определяет тип данных для данного атрибута.

Реляционные базы данных поддерживают операции CRUD (create, read, update, delete) для работы с данными, а также операции JOIN и GROUP BY для связывания данных из разных таблиц.

Данный блок представляет из себя модель данных или же реляционную базу данных, ее схему можно посмотреть на чертеже ГУИР.400201.307 РР2.

2.2 Блок взаимодействия с базой данных

Блок взаимодействия с базой данных – это важный компонент веб-приложения, который отвечает за связь между приложением и БД. Он позволяет сохранять, изменять и получать данные из БД.

Spring Data JPA является модулем Spring, который предоставляет удобный способ работы с базой данных через Java Persistence API (JPA). Он позволяет создавать репозитории для моделей данных приложения, которые автоматически будут преобразовываться в SQL-запросы, не требуя явного написания запросов. Spring Data JPA поддерживает различные реляционные базы данных, включая PostgreSQL, и предоставляет широкий спектр возможностей для работы с данными.

В случае данного дипломного проекта, данные будут передаваться в формате JSON. Spring Data JPA предоставляет возможность преобразовывать данные в этот формат для сохранения в БД и обратно для получения данных из БД. Это обеспечивает более простое и удобное взаимодействие между приложением и БД.

Spring Data JPA также позволяет использовать различные стратегии загрузки данных из БД. Жадная загрузка (eager loading) загружает все связанные объекты сразу при загрузке основного объекта, что может привести к увеличению нагрузки на БД и задержкам в работе приложения. Ленивая загрузка (lazy loading), напротив, загружает связанные объекты только при обращении к ним, что позволяет оптимизировать работу приложения и уменьшить нагрузку на БД. Она также позволяет создавать динамические запросы на основе критериев. Это означает, что запросы формируются на основе условий, заданных во время выполнения приложения, что упрощает процесс написания запросов на языке SQL и делает код более читаемым и понятным.

2.3 Блок пользовательского интерфейса

Блок пользовательского интерфейса будет написан на React - библиотеке JavaScript для разработки интерфейсов. В интерфейсе будут использоваться компоненты, которые позволяют легко создавать и управлять элементами интерфейса.

Для создания стилей и макетов интерфейса будет использоваться CSS. В React будет применяться подход "однонаправленного потока данных"

(unidirectional data flow), который позволяет управлять состоянием приложения через единственную точку входа.

Для обработки событий пользовательского взаимодействия и отправки запросов на сервер будет использоваться библиотека Axios. Для управления состоянием интерфейса будет применяться библиотека Redux, которая позволяет управлять состоянием приложения в централизованном месте и обеспечить предсказуемость работы интерфейса.

В будущем время мы будем использовать React Hooks, которые позволяют управлять состоянием компонентов и обрабатывать события без использования классовых компонентов. Также будем использовать React Router для управления маршрутизацией приложения и обеспечения SPA (Single Page Application) взаимодействия пользователей с интерфейсом.

2.4 Блок взаимодействия пользовательского интерфейса с контроллерами

В вышеописанном приложении клиентская часть будет взаимодействовать с серверной частью через REST API, который будет предоставлять контроллеры приложения.

При отправке запроса с клиента на сервер, запрос будет сначала проходить через маршрутизатор в React приложении, который определит адрес и метод запроса. Затем запрос будет отправлен на соответствующий адрес на сервере.

На серверной стороне запрос будет обрабатываться контроллером, который будет содержать логику обработки запроса и взаимодействия с базой данных. Контроллер будет принимать запрос, выполнять нужные действия, получать или отправлять данные в базу данных через репозиторий, и возвращать ответ клиенту в виде JSON объекта.

Клиентская часть приложения будет принимать ответ от сервера и отображать его на странице при помощи React компонентов. Если необходимо отправить данные на сервер, клиентская часть будет формировать объект с данными и отправлять его на сервер через API. В свою очередь, серверная часть будет обрабатывать запрос и сохранять данные в базу данных.

2.5 Блок контроллеров

Блок контроллеров в приложении будет отвечать за обработку HTTP запросов от клиента и передачу данных в сервисный слой приложения. Контроллеры будут реализованы на языке Java с использованием фреймворка Spring.

В приложении будут реализованы следующие контроллеры:

1. Контроллер для авторизации и аутентификации пользователей. Он будет принимать POST запросы на URL «`/api/auth/login`» и «`/api/auth/register`» для входа и регистрации пользователей

соответственно. Контроллер будет проверять корректность введенных пользователем данных и возвращать JWT токен в случае успешной аутентификации.

2. Контроллер для работы с сотрудниками. Он будет обрабатывать запросы на URL «`/api/employees`» для получения списка сотрудников. Контроллер позволит фильтровать сотрудников по определенным критериям, например по отделу и тому подобное.

3. Контроллер для работы с задачами. Он будет обрабатывать запросы на URL «`/api/tasks`» для получения списка задач, для авторизированного сотрудника. Также «`/api/tasks`» для создания новой задачи.

4. Контроллер для работы с событиями будет обрабатывать запросы на URL «`/api/events`» для получения списка событий для авторизированного пользователя. Также можно будет создавать разные события как для одного сотрудника, так и для разных групп, например по отделу, должности и тому подобному.

Каждый контроллер будет иметь свои методы для обработки различных HTTP запросов, таких как GET, POST, PUT и DELETE. Контроллеры будут также использовать аннотации фреймворка Spring для обозначения путей URL и типов HTTP запросов, которые они обрабатывают. Формат всех передаваемых данных будет JSON.

2.6 Блок бизнес-логики

Блок бизнес-логики – это блок, который обрабатывает запросы, отправленные клиентом, в качестве серверной части будет использоваться фреймворк Spring с различными «подфреймворками», о которых упоминалось и ранее, а также в разделе обзора литературы. Он будет включать в себя:

1. Регистрацию и авторизацию пользователей: при регистрации нового пользователя будет производиться проверка на уникальность логина и пароля. При авторизации пользователей будут проверяться логин и пароль с помощью токена, и если они верны, то пользователь получит доступ к системе.

2. Администрирование аккаунта только пользователем со специальными правами доступа. Оно включает в себе манипуляцию данными, а также возможность блокировать аккаунт.

3. Создание заданий и просьб: пользователи смогут создавать задачи и просьбы для других пользователей. При этом система будет автоматически назначать задания на определенных сотрудников и отслеживать наличие их у себя на странице.

4. Управление документами: пользователи смогут создавать и хранить документы в системе, а также иметь возможность отправлять их прямиком своим начальникам, а они в свою очередь смогут их подтверждать или опровергать. Это поможет облегчить взаимодействие и поможет сотрудникам выходить на больничный или в отпуск без необходимости ходить прямиком к начальнику. Также благодаря этому можно будет создать такую функцию, как

отслеживание сотрудников на работе. Если будет поступать подобное заявление и одобряться со стороны сотрудника, то на личной странице сотрудника будет отображаться его отсутствие, а также аккаунт будет временно заблокирован системой для обеспечения безопасности.

5. Управление событиями: в этом блоке будет реализована возможность создания событий, а также их назначения на других пользователей.

6. Организация доступа: система будет предоставлять два уровня доступа – администратор и пользователь. Каждый уровень будет иметь свои права доступа к определенным функциям приложения.

7. Оповещения: пользователи будут получать оповещения о новых заданиях, просьбах, событиях и других событиях в системе.

8. Фильтрация данных: система будет предоставлять функции фильтрации данных, чтобы пользователи могли быстро находить нужные сотрудников, документы, задания и т.д.

9. Работа с сотрудниками: приложение будет предоставлять возможность создания профиля для каждого сотрудника компании, хранения и управления персональной информацией, а также управления их задачами и присутствием на работе.

10. Возможность для каждого аккаунта сменить пароль.

11. При создании все пользователи по умолчанию будут создаваться как обычные пользователи. Возможность стать администратором может быть либо назначена с помощью другого администратора, либо напрямую через базу данных. Второй вариант не приветствуется из-за безопасности, но в чрезвычайных случая имеет место быть.

В целом, блок бизнес-логики будет включать в себя все функции, необходимые для эффективной организации работы сотрудников университета, управлению их задачами и документами и организации их коммуникации.

2.7 Блок авторизации пользователей

Блок авторизации пользователей в веб-приложении – это критически важный блок, который обеспечивает безопасность доступа к системе. В данном проекте для реализации авторизации будет использоваться Spring Security, который предоставляет нам множество инструментов для работы с безопасностью в приложениях.

В веб-приложении будет использоваться механизм авторизации на основе токенов JWT. Для этого настраивается специальный класс-конфигурация, в котором определяются правила доступа и их конфигурации. Также необходимо настроить процесс аутентификации, определить, какие поля будут использоваться для аутентификации, например, логин и пароль.

Далее создается сервис для работы с пользовательскими данными и методами для создания, чтения и обновления информации об аккаунтах пользователей.

Когда пользователь успешно проходит процесс аутентификации, то для него генерируется JWT токен, который он будет использовать для доступа к системе. Токен содержит информацию о пользователе и время его действия. Это удобно, так как можно настроить токены так, что по истечению, например, некоторого времени пользователю будет необходимо пройти авторизацию снова.

При каждом запросе на сервер мы будем проверять наличие токена в заголовке запроса и его валидность. Для этого создается и настраивается фильтр, который будет проверять токен и если он действителен, то пользователю разрешается доступ к запрашиваемому ресурсу.

Таким образом, блок авторизации пользователей в веб-приложении с использованием Spring Security и JWT токенов позволит обеспечить безопасность доступа к системе и защитить данные пользователей от несанкционированного доступа.

2.7 Блок USER

Блок пользователя в данном приложении предназначен для управления доступом к функционалу и предоставления возможности работать с данными. Для каждого пользователя в системе создается уникальный профиль, который содержит личные данные, такие как имя, фамилия, электронная почта, пароль, а также права доступа к функционалу приложения.

Пользователь может зарегистрироваться в системе и авторизоваться в ней, чтобы получить доступ к своей личной странице и функциям, которые ему разрешены в соответствии с его уровнем доступа.

Важной частью блока пользователя является система безопасности, которая обеспечивает защиту данных и доступа к функционалу приложения. Эта система включает в себя проверку подлинности пользователей, аутентификацию и авторизацию, а также защиту данных от несанкционированного доступа. Кроме того, внутри данной системы будет применяться система токенов для передачи информации о пользователе.

2.8 Блок ADMIN

Блок администратора в данном приложении предназначен для управления доступом к функционалу и предоставления возможности работать с данными. Благодаря специальным возможностям, правда у администраторов будут расширенные:

1. Управление учетными записями сотрудников: администратор сможет создавать, удалять и редактировать учетные записи сотрудников, а также изменять их уровень доступа и блокировать аккаунты, если по какой-то причине система дала сбой или необходимо заблокировать сотрудника из-за какой-то неординарной причины.
2. Мониторинг работы сотрудников: администратор сможет просматривать более углубленную информацию о сотрудниках.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Этот раздел посвящен описанию работы и состава разрабатываемого программного продукта.

В дальнейшем представлены взаимосвязи между различными классами программного обеспечения в виде диаграммы классов ГУИР.400201.307 РР.1.

3.1 Описание структуры приложения

Для разработки серверной части программного продукта были выбраны несколько технологий, которые используются в качестве инструментов и средств для создания функциональной и надежной системы. Эти технологии являются ключевыми элементами в разработке приложения и включают в себя набор инструментов для работы с базами данных, обеспечения безопасности и аутентификации пользователей, а также для реализации бизнес-логики приложения. Каждая из выбранных технологий имеет свои особенности и преимущества, что позволяет создавать более эффективную и гибкую систему в соответствии с требованиями проекта. Данные технологии в общих чертах рассматриваются ниже.

1. Spring Boot, которая позволяет создавать веб-сервер и настраивать взаимодействие между различными классами приложения, он автоматически добавляет в проект все необходимые зависимости и настраивает их для работы вместе.

2. Maven, используется для настройки процесса сборки, упаковки и запуска приложения, иначе говоря, для автоматизации сборки проектов. Данная технология использует файлы конфигурации POM (Project Object Model), в которых содержится информация о проекте, его зависимостях, конфигурациях, плагинах и других параметрах. С помощью данного инструмента и осуществляется подключение технологий Spring, например.

3. Spring Web, для создания веб-приложений. Этот модуль фреймворка Spring предоставляет инструменты для разработки на языке Java. Предоставляет ряд абстракций и компонентов, которые позволяют создавать масштабируемые, гибкие и безопасные веб-приложения.

4. Spring Data JPA, для работы с базой данных. Предоставляет реализацию JPA, которая упрощает доступ к базе данных и сокращает объем кода, необходимого для создания репозиториев и выполнения операций с базой данных. Spring Data JPA позволяет автоматически генерировать репозитории, которые позволяют выполнять CRUD (Create, Read, Update, Delete) операции с объектами, не нужно писать много кода вручную.

5. Spring Security, для обеспечения защиты и авторизации пользователей в системе. Предоставляет инструменты для обеспечения безопасности веб-приложений на основе Java.

В данном приложении серверная часть приложения реализовала на Java. Исходя из гексагональной архитектуры, в приложении классы делятся по

исполняемому функционалу. Вся структура пакетов и их описание приведена ниже (см. рисунок 3.1):

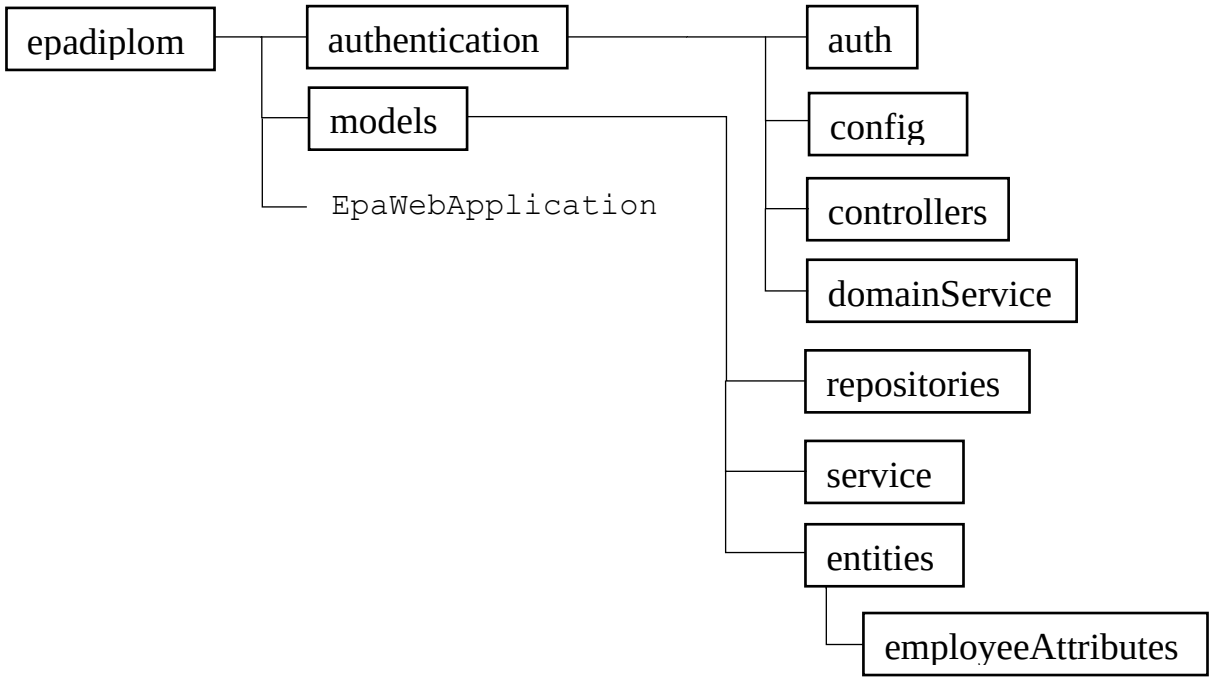

Рисунок 3.1 – Структура пакетов в веб-приложении

1. Пакет authentication – пакет предназначенный для реализации функционала, связанного с аутентификацией пользователей, включая создание аккаунтов, вход в систему и другие смежные задачи. Он содержит несколько подпакетов, которые обеспечивают логическую группировку функционала по его назначению и упрощают работу с ним. Эти подпакеты рассматриваются ниже:

1.1 Подпакет auth – пакет содержащий классы, которые отвечают за реализацию процесса аутентификации и авторизации пользователей в системе. Здесь находятся реализации функций создания аккаунтов, входа в систему и других задач, связанных с проверкой личности пользователя. Таким образом, данный пакет является важным компонентом системы безопасности и гарантирует правильный доступ к ресурсам системы только авторизованным пользователям. В него входят классы, перечисленные ниже:

- `authenticationRequest`;
- `authenticationResponse`;
- `authenticationService`;
- `registerRequest`.

1.2 Подпакет config –подпакет содержащий классы, которые отвечают за конфигурацию и настройку Spring Security, фреймворка, предназначенного для обеспечения безопасности приложений на платформе Spring. Данный пакет также отвечает за настройку JSON Web Token (JWT), механизма аутентификации и авторизации пользователей, использующего технологию

передачи данных в формате JSON. Здесь происходит создание и настройка токенов, которые используются для идентификации пользователей и обеспечения безопасного доступа к ресурсам системы. Благодаря настройке JWT в данном пакете, система гарантирует безопасность передачи данных между клиентом и сервером и защищает от несанкционированного доступа. В него входят классы, перечисленные ниже:

- `applicationConfig`;
- `jwtAuthentificationFilter`;
- `securityConfig`.

1.3 Подпакет controllers – подпакет содержащий классы-контроллеры, которые являются частью паттерна проектирования Model-View-Controller (MVC). Контроллеры представляют собой классы, которые обрабатывают запросы от клиента и выполняют соответствующие действия в системе. Внутри каждого контроллера находятся методы-обработчики, которые реагируют на определенный тип запросов и возвращают клиенту соответствующий ответ. Данный пакет играет важную роль в обработке запросов и представляет собой основной механизм, с помощью которого клиент взаимодействует с системой. Содержит в себе классы:

- `authenticationController`;
- `mainPageConroller`.

1.4 Подпакет domainsService – подпакет, с помощью которого происходит управление JSON Web Token (JWT), реализуемом в данном приложении, для создания своеобразных ключей, которые помогают при взаимодействии клиента и сервера. Содержит в себе класс:

- `jwtService`.

2. Пакет models – пакет, представляющий основную модель данных системы, и содержит все компоненты, связанные с управлением сущностями и взаимодействием с базой данных. В нем находятся несколько подпакетов, отвечающих за доступ к данным, их обработку и сохранение в базе данных. Этот пакет можно назвать прослойкой между моделью данных и базой данных, так как он обеспечивает связь между ними. Кроме того, данный пакет является ключевым компонентом системы, так как представляет основную модель данных и определяет структуру и взаимосвязи между сущностями.

2.1 Подпакет entities – подпакет содержащий сущности базы данных, которые используются вместе с подпакетами service и repositories для реализации бизнес-логики приложения. Он включает в себя 16 классов, из которых четыре являются представлениями, которые уже были упомянуты в модели данных и не нуждаются в дополнительном описании. Сущности, находящиеся в этом пакете, служат основой для работы с базой данных и представляют структуру данных, которые хранятся в ней. Вместе с классами из пакетов service и repositories, этот пакет обеспечивает полную реализацию бизнес-логики приложения.

2.1.1. Пакет в подпакете employeeAttribures содержит список

именованных констант – `role`, в котором находятся роли пользователей. Этот список используется для управления уровнем доступа пользователей при аутентификации и авторизации в системе. Различные роли дают пользователям различные уровни доступа.

2.2 Подпакет krpositories – подпакет содержащий интерфейсы, которые используются для взаимодействия с базой данных с помощью репозиториев, таких как JpaRepository. Это упрощает создание запросов к таблицам и в сочетании с пакетами service и entities позволяет реализовывать бизнес-логику приложения.

2.3 Подпакет service – подпакет содержащий классы, связанные с сервисом сущностей таблицы. С помощью этих классов можно реализовать сложные запросы к таблицам, которые не могут быть выполнены с помощью JpaRepository. В сочетании с пакетами repositories и entities, они позволяют реализовать бизнес-логику приложения.

Отдельно от всех этих моделей и пакетов находится класс `EpaWebApplication` в общей папке со всем вышеперечисленным с названием `epadiplom`.

Подобная организация пакетов в приложении позволяет удобно добавлять новый функционал и вносить изменения в существующий код. При создании нового компонента можно просто создать новый пакет и добавить в него нужные классы, не затрагивая другие компоненты приложения. Такая организация обеспечивает изоляцию функционала и позволяет легко поддерживать код приложения. Кроме того, такая структура приложения упрощает работу команды разработчиков и ускоряет процесс разработки.

3.2 Описание модели данных

В данном разделе рассматривается база данных, которая работает с помощью СУБД PostrgeSQL. Этот блок включает в себя данные, которые использует разрабатываемая система.

Для удобства модель данных можно условно разбить на несколько логических блоков, каждый из которых будет выполнять свою функцию в создаваемом веб-приложении (см. рисунок 3.2).

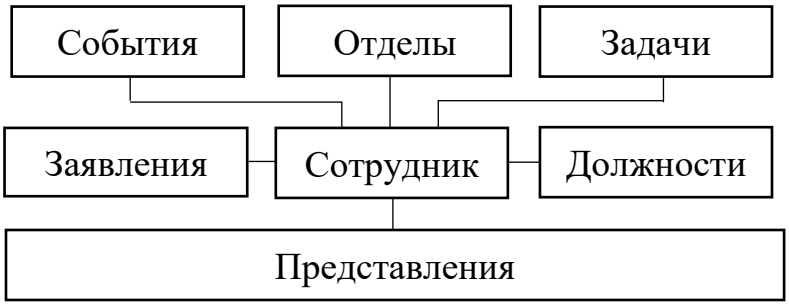

Рисунок 3.2 – Условное логическое разделение блоков БД

Данные сотрудника будут использоваться как для основной информации о сотруднике, так и хранить данные о пользователе, которые помогут получать информацию об определенном сотруднике.

Данные по событиям, задачам, отделам, заявлениям и должностям помогут получать информацию, исходя из их названий.

Все вышеперечисленные блоки представляют собой таблицы реляционной базы данных, которыми можно всячески манипулировать, но, в представленном логическом разделении, также присутствует блок представлений.

Представления являют собой виртуальные таблицы, которые будут помогать облегчать доступ к данным, не прибегая к созданию сложных запросов. Они нужны, если необходимо получить информацию из нескольких таблиц одновременно. Такие «таблицы» можно только просматривать без изменения данных. Чтобы их изменить придется обращаться к исходным таблицам.

3.2.1 Таблица `Employee`

Данная таблица предназначена для хранения основной информации о пользователе, которая можно указать в общем доступе, и которая будет отображаться в глобальном поиске сотрудников.

Поля таблицы:

- `id` – первичный ключ, bigint;
- `first_name` – имя сотрудника, varchar (128);
- `middle_name` – отчество сотрудника, varchar (128);
- `last_name` – фамилия сотрудника, varchar (128);
- `work_number` – рабочий номер сотрудника, numeric (16);
- `location_street` – зашехированный пароль, который был выслан пользователю для восстановления аккаунта, varchar (128);
- `cabinet_office` – время и дата отправления пароля для восстановления аккаунта, varchar(8);
- `id_dep` – внешний ключ для связи с таблицей `department`, bigint.

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. С полем `id` используется специальный атрибут «null: false», это означает, что поле не может быть нулевым. Для остальных колонок это значение не выставлено, чтобы при создании аккаунта для сотрудника данные заполнялись отделом кадров и не было ошибок в системе и документах.

3.2.2 Таблица `Login`

Данная таблица предназначена для хранения информации о пользователе, которая связана с аккаунтом сотрудника, а именно содержит данные необходимые для авторизации и создания аккаунта.

Поля таблицы:

- id_login – первичный ключ, а также внешний ключ для таблицы employee, bigint;
- login_user – логин сотрудника, varchar (319);
- password_user – захешированный пароль, varchar (128);
- mail_user – мейл сотрудника, varchar (319);
- role – роль пользователя (касаемо аккаунта, а не работы), varchar (6).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. С полем id_login, login_user, password_user, mail_user, role используется специальный атрибут «null:false», это означает, что поле не может быть нулевым. Все эти поля заполняются при создании профиля сотрудника.

3.2.3 Таблица Personal

Данная таблица предназначена для хранения личной информации о пользователе, которую можно будет увидеть только при наличии определенных прав, и которая требуется, в основном, отделу кадров.

Поля таблицы:

- id_personal – первичный ключ, а также внешний ключ для таблицы employee, bigint;
- birth_d – дата рождения сотрудника, timestamp without time zone;
- entry_d – дата устройства на работу (в этот же день должен быть и создан аккаунт), timestamp without time zone;
- personal_number – личный номер сотрудника, numeric (16).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. С полем id_personal используется специальный атрибут «null: false», это означает, что поле не может быть нулевым. Для остальных колонок это значение не выставлено, чтобы при создании аккаунта для сотрудника данные заполнялись отделом кадров и не было ошибок в системе и документах. Поле entry_d будет заполняться автоматически при создании аккаунта. Формат данных timestamp without time zone означает, что нет привязки к часовому поясу. Это сделано, чтобы не было разногласий во времени.

3.2.4 Таблица Department

Данная таблица нужна для содержания списка отделов университета, чтобы было удобнее распределять и сортировать сотрудников.

Поля таблицы:

- id – первичный ключ, bigint;
- name_dep – название отдела университета, varchar (128).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. С полями id и name_dep используется специальный атрибут «null: false». Эти поля не могут быть нулевыми.

3.2.5 Таблица Log_statement

Данная таблица предназначена для информации о заявлениях. Они могут быть разных типов: от отпусков, до увольнения за свой счет и тому подобные.

Поля таблицы:

- id – первичный ключ, bigint;
- days_sum – сумма дней, integer;
- date_leave – дата, когда работник уходит по заявлению (дата действия заявления), timestamp without time zone;
- date_of_ls – дата, когда работник составляет заявление, timestamp without time zone;
- id_approver – номер сотрудника, который должен подтвердить заявление, bigint;
- comment_ls – комментарий сотрудника к заявлению, varchar (300);
- type_leave – тип заявления (типы будут прописаны в логике), smallint;
- approve – статус подтверждения, numeric;
- id_employee – внешний ключ для таблицы employee (того работника, что составляет заявление), bigint.

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля за исключением comment_ls используется специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении и сотрудник не может оставить их пустыми, за исключением комментария и самого скана документа. Поле date_of_ls будет заполняться автоматически при создании аккаунта. Формат данных timestamp without time zone означает, что нет привязки к часовому поясу. Это сделано, чтобы не было разногласий во времени.

3.2.6 Таблица Document

Данная таблица предназначена для хранения сканов оригинальных заявлений. Они не являются обязательными, поэтому таблица log_statement может существовать без привязки к данной таблице.

Поля таблицы:

- id – первичный ключ, bigint;
- id_ls – внешний ключ для таблицы log_statement, bigint;
- body_doc – ссылка на скан оригинального заявления, varchar (300).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля используют специальный

атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении.

3.2.7 Таблица `Job_title`

Данная таблица предназначена для хранения списка должностей сотрудников БГУИР.

Поля таблицы:

- `id` – первичный ключ, bigint;
- `job_title_name` – название должности, varchar (128).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля используют специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении.

3.2.8 Таблица `Job_employee`

Данная таблица является реализацией связи многие-ко-многим между таблицами `job_title` и `employee`.

Поля таблицы:

- `id` – первичный ключ, bigint;
- `id_job_title` – внешний ключ для таблицы `job_title`, bigint;
- `id_employee` – внешний ключ для таблицы `employee`, bigint.

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля используют специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении.

3.2.9 Таблица `Task`

Данная таблица предназначена для хранения информации о заданиях или же действиях, которые нужно сделать работнику.

Поля таблицы:

- `id` – первичный ключ, bigint;
- `date_task` – дата, когда работник составляет заявление, timestamp without time zone;
- `name_of_task` – название задания или его суть, varchar (128);
- `id_executor` – номер сотрудника, который будет исполнять задание, bigint;
- `comment_te` – комментарий сотрудника к заданию, по сути, описание, если таковое требуется, varchar (300).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля, кроме `comment_te`, используют специальный атрибут «null: false», это означает, что поле не может

быть нулевым, потому что все эти поля имеют важное значение при заполнении. Поле comment_te можно пропустить, потому что некоторые задания могут быть ясны без уточнений. Формат данных timestamp without time zone означает, что нет привязки к часовому поясу. Это сделано, чтобы не было разногласий во времени.

3.2.10 Таблица Emp_task

Данная таблица является реализацией связи многие-ко-многим между таблицами task и employee.

Поля таблицы:

- id – первичный ключ, bigint;
- id_job_title – внешний ключ для таблицы job_title, bigint;
- id_employee – внешний ключ для таблицы employee, bigint.

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля используют специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении.

3.2.11 Таблица Event

Данная таблица предназначена для хранения информации о событиях, созданных сотрудниками, а также назначения, для кого они предназначены.

Поля таблицы:

- id – первичный ключ, bigint;
- date_of_event – дата события, timestamp without time zone;
- type_of_event – название события, varchar (40);
- comment_fe – комментарий сотрудника к событию, по сути, описание, если таковое требуется, varchar (300).

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля, кроме comment_fe, используют специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении. Поле comment_fe можно пропустить, потому что некоторые задания могут быть ясны без уточнений. Формат данных timestamp without time zone означает, что нет привязки к часовому поясу. Это сделано, чтобы не было разногласий во времени.

3.2.12 Таблица Notice_event

Данная таблица является реализацией связи многие-ко-многим между таблицами event и employee.

Поля данной таблицы:

- id – первичный ключ, bigint;

- id_event – внешний ключ для таблицы event, bigint;
- id_recipient – номер сотрудника, которому предназначается отправить событие, bigint;
- id_employee – внешний ключ для таблицы employee, bigint.

В таблице для колонок существуют определенные атрибуты и условия, которые задавались при их создании. Все поля используют специальный атрибут «null: false», это означает, что поле не может быть нулевым, потому что все эти поля имеют важное значение при заполнении.

3.2.13 Представление Employee_full_info_view

Данная таблица является представлением, виртуальной или же логической таблицей, которая представляет собой поименованный запрос. Это представление нужно для облегченной работы с блоком данных, касаемых общей информации о сотруднике, которая представляет собой данные по аккаунту, персональные данные и общие сведения, которые будут в общем доступе.

Таблицы, которые объединены в данном представлении: personal, employee, department, job_employee, job_title, login.

Поля представления:

- job_employee.id;
- login.id_login;
- employee.first_name;
- employee.middle_name;
- employee.last_name;
- personal.birth_d;
- personal.entry_d;
- login.login_user;
- login.password_user;
- login.mail_user;
- login.role;
- employee.work_number;
- personal.personal_number;
- employee.location_street;
- employee.cabinet_office;
- department.name_dep;
- job_title.job_title_name.

3.2.14 Представление Employees_view

Данная таблица является представлением, виртуальной или же логической таблицей, которая представляет собой поименованный запрос. Это представление нужно для облегченной работы с блоком данных, касаемых

общей информации о сотруднике, которые могут быть в общем доступе и видимы для других сотрудников.

Таблицы, которые объединены в данном представлении: `employee`, `department`, `job_employee`, `job_title`.

Поля представления:

- `job_employee.id`;
- `job_employee.id_employee`;
- `employee.first_name`;
- `employee.middle_name`;
- `employee.last_name`;
- `employee.work_number`;
- `employee.location_street`;
- `employee.cabinet_office`;
- `department.name_dep`;
- `job_title.job_title_name`.

3.2.15 Представление `Ls_view`

Данная таблица является представлением, виртуальной или же логической таблицей, которая представляет собой поименованный запрос. Это представление нужно для облегченной работы с блоком данных, касаемых заявлений и их заполнения.

Таблицы, которые объединены в данном представлении: `employee`, `log_statement`, `document`, `login`.

Поля представления:

- `document.id`;
- `log_statement.type_leave`;
- `log_statement.date_leave`;
- `log_statement.date_of_ls`;
- `log_statement.days_sum`;
- `log_statement.id_approver`;
- `log_statement.approve`;
- `log_statement.comment_ls`;
- `log_statement.id_employee`;
- `document.body_doc`;
- `login.role`.

3.2.16 Представление `Events_view`

Данная таблица является представлением, виртуальной или же логической таблицей, которая представляет собой поименованный запрос. Это представление нужно для облегченной работы с блоком данных, касаемых событий, касаемых сотрудников университета.

Таблицы, которые объединены в данном представлении: `event`, `notice_event`. Поля представления:

- `notice_event.id`;
- `event.type_of_event`;
- `event.date_of_event`;
- `event.comment_fe`;
- `notice_event.id_recipient`;
- `notice_event.id_employee`.

3.2.17 Представление `Job_title_view`

Данная таблица является представлением, виртуальной или же логической таблицей, которая представляет собой поименованный запрос. Это представление нужно для облегченной работы с блоком данных, касаемых должности сотрудников.

Таблицы, которые объединены в данном представлении: `job_title`, `job_employee`.

Поля представления:

- `job_employee.id`;
- `job_title.job_title_name`;
- `job_employee.id_employee`.

3.3 Описание структуры и взаимодействия между классами

При создании приложения использовался паттерн Model-View-Controller (MVC), который определяет его структуру, состоящую из трех основных компонентов: контроллера, сервиса и репозитория. Кроме того, следует отметить, что все созданные сервисы разработаны в соответствии с правилами и стандартами REST-архитектуры. Это касаемо серверной части приложения.

Контроллеры отвечают за обработку входящих HTTP-запросов и вызывают соответствующие методы сервисов, которые обрабатывают эти запросы и возвращают результаты. Сервисы представляют собой прослойку между контроллером и репозиторием и отвечают за бизнес-логику приложения, такую как проверка прав доступа, обработка данных и т.д. Репозитории служат для связи с базой данных и содержат методы для выполнения CRUD-операций (создание, чтение, обновление и удаление данных).

3.3.1 Класс `EpaWebApplication`

Точка входа для запуска веб-приложения на основе фреймворка Spring Boot. Аннотация `@SpringBootApplication` указывает, что это главный класс

приложения и сообщает Spring, что нужно выполнить все необходимые конфигурации и инициализации для запуска веб-приложения.

Метод `main()` вызывает метод `run()` класса `SpringApplication`, который запускает приложение. В качестве аргументов метод `run()` принимает класс `EpaWebApplication` и аргументы командной строки `args`.

Таким образом, этот класс и его метод `main()` запускают Spring Boot приложение и начинают обработку входящих HTTP запросов.

3.3.2 Класс `AuthenticationRequest`

Этот класс представляет собой модель данных для запроса аутентификации пользователя в системе.

В данном классе используются аннотации фреймворка Lombok – `@Data`, `@Builder`, `@AllArgsConstructor`, `@NoArgsConstructor`. Они позволяют автоматически генерировать стандартные методы JavaBean, такие как `toString()`, `equals()`, `hashCode()` и геттеры/сеттеры для всех полей класса, что сокращает количество необходимого для написания кода.

Класс определяет структуру запроса на аутентификацию пользователя в системе и используется для передачи данных между клиентским и серверным приложениями. Имеет поля `private String login` и `String password`.

Эти поля нужны для представления данных, передаваемых для аутентификации пользователя.

3.3.3 Класс `AuthenticationResponce`

Этот класс представляет собой модель данных для ответа на запрос аутентификации пользователя в системе.

В данном классе используются аннотации фреймворка Lombok – `@Data`, `@Builder`, `@AllArgsConstructor`, `@NoArgsConstructor`. Они позволяют автоматически генерировать стандартные методы JavaBean, такие как `toString()`, `equals()`, `hashCode()` и геттеры/сеттеры для всех полей класса, что сокращает количество необходимого для написания кода.

Класс, как и класс, приведенный выше, также определяет структуру ответа на запрос аутентификации пользователя в системе и используется для передачи данных между серверным и клиентским приложениями.

Имеет всего одно поле:

- `private String token`.

Токен является строкой, которая используется для идентификации пользователя на сервере и доступа к защищенным ресурсам.

3.3.4 Класс `AuthenticationService`

Этот класс представляет собой сервис, который предоставляет функциональность регистрации и аутентификации пользователей в системе.

Аннотация `@Service` указывает, что этот класс является сервисом и должен быть управляемым Spring контейнером.

Класс имеет четыре поля, приведенных ниже:

- `private final UserRepo userRepo`;
- `private final PasswordEncoder passwordEncoder`;
- `private final JwtService jwtService`;
- `private final AuthenticationManager authenticationManager`.

Методы класса:

- `register()` выполняет регистрацию нового пользователя в системе;
- `authenticate()` выполняет аутентификацию пользователя в системе.

Таким образом, этот класс предоставляет функциональность регистрации и аутентификации пользователей в системе, используя Spring Security и JSON Web Token (JWT).

3.3.5 Класс `RegisterRequest`

Класс используется как часть процесса регистрации нового пользователя в системе. По сути, этот класс является моделью данных (data model), представляющей структуру запроса на регистрацию нового пользователя. В данном классе используются аннотации фреймворка Lombok – `@Data`, `@Builder`, `@AllArgsConstructor`, `@NoArgsConstructor`. Они позволяют автоматически генерировать стандартные методы JavaBean, такие как `toString()`, `equals()`, `hashCode()` и геттеры/сеттеры для всех полей класса, что сокращает количество необходимого для написания кода.

Имеет поля, приведенные ниже:

- `private String firstName`;
- `private String password`;
- `private String mail`.

3.3.6 Класс `ApplicationConfig`

Класс представляет собой конфигурационный класс Spring, который содержит конфигурацию для аутентификации пользователей в системе. В классе определены следующие методы, описанные ниже:

1. Класс `userDetailsService` возвращает сервис для поиска пользователей по имени пользователя, используя репозиторий `UserRepo`.

2. Класс `authenticationProvider` создает провайдера аутентификации `DaoAuthenticationProvider`, который использует `userDetailsService` для поиска пользователя в базе данных и `passwordEncoder` для проверки пароля пользователя.

3. Класс `authenticationManager` создает и возвращает менеджер аутентификации `AuthenticationManager`, используя конфигурацию аутентификации.

4. Класс `passwordEncoder` возвращает объект `BCryptPasswordEncoder`, который используется для хэширования пароля пользователя.

3.3.7 Класс `JwtAuthenticationFilter`

Данный код представляет собой фильтр аутентификации, который будет вызван один раз для каждого запроса, прошедшего через контроллер в приложении. Фильтр проверяет наличие токена авторизации в заголовке запроса и, если он присутствует, использует сервис JWT для проверки его валидности и получения имени пользователя из токена. Затем фильтр проверяет, что пользователь существует в базе данных и, если это так, создает аутентификационный токен Spring Security и устанавливает его в контекст безопасности. Если токен авторизации не найден или недействителен, фильтр пропускает запрос и передает его дальше по цепочке фильтров.

В нем всего один метод, который выполняет все вышеперечисленное:

- `protected void doFilterInternal`;

Также в коде есть два поля, которые получает конструктор:

- `private final JwtService jwtService` – класс, который реализует логику работы с JWT токенами;
- `private final UserDetailsService userDetailsService` – сервис, который будет использоваться для загрузки информации о пользователе по логину.

3.3.8 Класс `SecurityConfig`

Этот класс содержит конфигурацию Spring Security для веб-приложения. Он использует аннотации Spring `@Configuration` и `@EnableWebSecurity`, чтобы сообщить Spring, что этот класс содержит конфигурацию безопасности для веб-приложения. Данный класс содержит метод:

- `securityFilterChain` – метод, который создает цепочку фильтров безопасности.

В данном классе еще есть два поля:

- `jwtAuthFilter` – это объект фильтра, который будет использоваться для проверки JWT-токенов и аутентификации пользователей;
- `authenticationProvider` – это объект, который будет использоваться для проверки учетных данных пользователей.

3.3.9 Класс `AuthenticationController`

Данный класс представляет контроллер для обработки запросов, связанных с аутентификацией и авторизацией пользователей.

Аннотация `@RestController` указывает на то, что класс предназначен для обработки HTTP-запросов, а возвращаемые им методы должны быть преобразованы в тело ответа HTTP.

Аннотация `@RequestMapping("/api/v1/auth")` указывает на корневой путь, который будет использоваться для обработки запросов, обрабатываемых этим контроллером.

В классе есть несколько методов, которые рассмотрены ниже:

1. Метод `register` – он обрабатывает POST-запросы на `/api/v1/auth/register`. Он принимает в теле запроса объект `RegisterRequest`, содержащий данные, необходимые для регистрации нового пользователя, и передает их в сервис `AuthenticationService` для выполнения регистрации. Затем он возвращает объект `AuthenticationResponse`, содержащий информацию об успешности регистрации и авторизации нового пользователя.

2. Метод `authenticate` – он обрабатывает POST-запросы на `/api/v1/auth/authenticate`. Он принимает в теле запроса объект `AuthenticationRequest`, содержащий учетные данные пользователя (имя пользователя и пароль), и передает их в сервис `AuthenticationService` для выполнения аутентификации. Затем он возвращает объект `AuthenticationResponse`, содержащий JWT-токен, который пользователь может использовать для авторизации на защищенных ресурсах.

3. Метод `sayHello` – он обрабатывает GET-запросы на `/api/v1/auth/authorization`. Он возвращает строку «`Hello from secured endpoint`», что означает успешное прохождение аутентификации и авторизации пользователем. Он используется для проверки работоспособности механизма аутентификации и авторизации.

3.3.10 Класс `MainPageController`

Этот класс является контроллером Spring Boot и содержит обработчики HTTP-запросов. Он предназначен для работы с главной страницей приложения. Класс MainPageController использует несколько репозиториев для доступа к данным в базе данных, которые хранят информацию о сотрудниках, логах, событиях и других объектах. Каждый метод возвращает список объектов, который сериализуется в JSON и отправляется обратно клиенту в ответ на запрос. Также в этом классе есть методы, которые используют Spring Security для аутентификации пользователей и контроля доступа к данным.

Содержит ряд методов, которые будут рассмотрены ниже:

1. Метод `getEmployeeInfo()` – этот метод контроллера происходит получение данных о залогиненном пользователе, которые содержатся в таблице `employee_full_view`. Метод `findAllByIdLogin()` выполняет выборку всех записей из этой таблицы для залогиненного пользователя.

2. Метод `getEmployees()` – этот метод возвращает список всех пользователей, зарегистрированных в системе. Запрос к базе данных выполняется с использованием метода `findAll()` из репозитория `employeesViewRepo`.

3. Метод `getLsRequests()` – этот метод получает список запросов на изменение данных (`log statements`), которые ожидают подтверждения со стороны пользователя. Выборка выполняется с использованием метода `findAllByIdApproverAndApprove()` из репозитория `logStatementViewRepo`. Параметр `idApprover` указывает на идентификатор пользователя, которому требуется подтверждение изменений, а `approve` задает статус запроса (1 - подтвержден, 2 - отклонен, 3 - требуется подтверждение).

4. Метод `getEvents()` – этот метод возвращает список всех событий, связанных с пользователем. Запрос выполняется с использованием метода `findAllByIdRecipient()` из репозитория `eventsViewRepo`. Параметр `idRecipient` указывает на идентификатор пользователя, для которого запрашиваются события.

3.3.11 Класс `JwtService`

Этот класс предоставляет функционал для генерации и проверки JSON Web Tokens (JWT), которые используются для аутентификации пользователей в приложениях.

Данный класс содержит следующие методы:

- `extractUsername(jwToken)` – извлекает имя пользователя из JWT;
- `extractClaim(jwToken, claimsResolver)` – извлекает любое утверждение из JWT, используя переданный функциональный интерфейс `claimsResolver`;
- `generateToken(userDetails)` – генерирует JWT для пользователя `userDetails`;
- `isTokenValid(jwToken, userDetails)` – проверяет, действителен ли JWT для пользователя `userDetails`;
- `isTokenExpired(jwToken)` – проверяет, истекло ли время жизни JWT;
- `extractExpiration(jwToken)` – извлекает дату истечения срока действия JWT;
- `extractAllClaims(jwToken)` – извлекает все утверждения из JWT;
- `generateToken(extraClaims, userDetails)` – генерирует JWT с переданными дополнительными утверждениями `extraClaims` для пользователя `userDetails`.

Для работы с JWT используется библиотека JSON Web Token (io.jsonwebtoken) и алгоритм подписи HS256 (используется ключ, заданный в поле `SECRET_KEY`).

3.3.12 Перечисление `Role`

Данное перечисление представляет собой список возможных ролей пользователей системы, которые могут быть назначены сотрудникам.

Константы ADMIN и USER определяют две роли: администратор и обычный пользователь.

3.3.13 Классы сущностей

Далее идут классы сущностей, описывающих таблицы в базу данных. Они схожи по структуре и содержанию. Аннотация @Entity сообщает JPA, что данный класс является сущностью, которая будет отображаться на таблицу в базе данных. Аннотация @Table используется для указания имени таблицы, аннотация @NoArgsConstructor генерирует конструктор без параметров, а аннотация @Getter генерирует геттеры для всех полей класса. Также в классах описываются связи между таблицами с помощью аннотаций @OneToMany, @ManyToOne и @OneToOne. Ниже перечислены классы-сущности:

- класс Department;
- класс Document;
- класс Employee;
- класс EmployeeFullView;
- класс EmployeesView;
- класс EmployeeTask;
- класс Event;
- класс EventsView;
- класс JobEmployee;
- класс JobTitle;
- класс LogStatement;
- класс LogStatementsView;
- класс NoticeEvent;
- класс Personal;
- класс Task;
- класс User.

Подробнее рассмотрим сущность User. Она представляет собой таблицу Login из базы данных. Данная сущность нужна для реализации авторизации пользователя в системе, потому что позволяет манипулировать данными сотрудника и в принципе позволяет осуществлять связь конкретного пользователя с базой данных.

Класс User также реализует интерфейс UserDetails из Spring Security, который содержит методы для получения информации о пользователе, используемой при аутентификации и авторизации пользователей в приложении.

В частности, методы getAuthorities(), getPassword(), getUsername() используются для проверки прав доступа пользователей в системе, а методы isAccountNonExpired(), isAccountNonLocked(), isCredentialsNonExpired(), isEnabled() позволяют проверять статусы учетной записи пользователя.

Также, этот класс определяет соответствующие поля и методы для работы с базой данных, используя аннотации JPA.

3.3.14 Интерфейсы репозиториев

Теперь рассмотрим интерфейсы, являющиеся репозиториями для сущностей, описанных ранее. Они наследуются от `JpaRepository<Repo, Long>`, что позволяет ему использовать стандартные методы доступа к данным (CRUD), такие как сохранение, обновление, удаление, поиск и так далее. Далее перечислены все интерфейсы для классов-сущностей:

- интерфейс `DepartmentRepo`;
- интерфейс `DocumentRepo`;
- интерфейс `EmployeeFullViewRepo`;
- интерфейс `EmployeeRepo`;
- интерфейс `EmployeesViewRepo`;
- интерфейс `EmployeeTaskRepo`;
- интерфейс `EventRepo`;
- интерфейс `EventsViewRepo`;
- интерфейс `JobEmployeeRepo`;
- интерфейс `JobTitleRepo`;
- интерфейс `LogStatementRepo`;
- интерфейс `LogStatementsViewRepo`;
- интерфейс `NoticeEventRepo`;
- интерфейс `PersonalRepo`;
- интерфейс `TaskRepo`;
- интерфейс `UserRepo`.

3.3.15 Классы сервиса

Теперь рассмотрим классы сервиса. Эти классы являются слоем сервиса и отвечает за бизнес-логику, связанную с сущностями из подраздела 3.3.13. В них содержатся методы, которые могут вызываться из контроллеров для обработки запросов, связанных с их сущностями, такие как создание, чтение, обновление и удаление, а также другие методы, связанные с бизнес-логикой. Сервисный слой использует репозиторий для доступа к данным и предоставляет абстракцию и контроль над данными, что позволяет легче модифицировать бизнес-логику и масштабировать приложение. Далее будут представлен список всех классов, используемых для сервиса: `DepartmentService`, `DocumentService`, `EmployeeService`, `EmployeeTaskService`, `EventService`, `JobEmployeeService`, `JobTitleService`, `LogStatementService`, `NoticeEventService`, `PersonalService`, `TaskService`, `UserService`.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Sage HR [Электронный ресурс]. – Электронные данные. – Режим доступа: https://sage.hr/ru. – Дата доступа: 01.04.2023.

[2] WebHR [Электронный ресурс]. – Электронные данные. – Режим доступа: https://web.hr. – Дата доступа: 01.04.2023.

[3] Архитектура веб-приложений: принципы, протоколы, практика. / Л. Шкляр Р. Розен – Эксмо, 2011. – 634 с.

[4] Клиент-серверная архитектура [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.hexlet.io/courses/internet-fundamentals/lessons/client-server/theory_unit. – Дата доступа: 01.04.23.

[5] Трехуровневая клиент-серверная архитектура [Электронный ресурс]. – Электронные данные. – Режим доступа: https://testmatick.com/ru/osnovnye-ponyatiya-i-osobennosti-klient-servernoj-arhitektury/. – Дата доступа: 01.04.23.

[6] Паттерны объектно-ориентированного проектирования. / Э. Гамма Р. Хелм Р. Джонсон Дж. Влиссидес – СПб. Издательство Питер, 2020. – 448с.

[7] Принцип SOLID [Электронный ресурс]. – Электронные данные. – Режим доступа: https://web-creator.ru/articles/mvc. – Дата доступа: 01.04.23.

[8] PostgreSQL [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.postgresql.oërg/. – Дата доступа: 01.04.2023.

[9] Spring Framework [Электронный ресурс]. – Электронные данные. – Режим доступа: https://spring.io/why-spring. – Дата доступа: 01.04.23.

[10] React [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.reactjs.org. – Дата доступа: 01.04.23.

ПРИЛОЖЕНИЕ А

(обязательное)

Вводный плакат

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема структурная

ПРИЛОЖЕНИЕ В

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Г

(обязательное)

Модель данных