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.

2.8 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.

2.9 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.