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, что 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 запросов;
- блок бизнес-логики;
- блок авторизации пользователей;
- блок user;
- блок admin;
- блок block.

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. Эта база данных поддерживает Atomicity, Consistency, Isolation, Durability или же ACID-принципы, 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). Этот модуль 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 передаются в формате JSON. Spring Data JPA предоставляет возможность преобразовывать данные в этот формат для сохранения в БД и обратно для получения данных из БД, что более просто и удобно позволяет взаимодействовать приложению и БД.

Spring Data JPA также позволяет использовать различные стратегии загрузки данных из БД. Жадная загрузка загружает все связанные объекты сразу при загрузке основного объекта, что может привести к увеличению нагрузки на БД и задержкам в работе приложения. Ленивая загрузка, напротив, загружает связанные объекты только при обращении к ним, что позволяет оптимизировать работу приложения и уменьшить нагрузку на БД. Такая загрузка 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 написан на React - библиотеке JavaScript для разработки интерфейсов. В интерфейсе используются компоненты, которые позволяют легко создавать и управлять элементами интерфейса.

Для создания стилей и макетов интерфейса используется CSS (Cascading Style Sheets) – это язык, используемый в веб-разработке для описания визуального представления документа, написанного на языке HTML

(Hypertext Markup Language). С помощью CSS можно определять стили элементов веб-страницы, такие как цвет, размер и расположение текста, фоновые изображения, отступы, рамки и т.д. CSS позволяет разделить содержимое документа и его представление, что позволяет создавать более гибкие и масштабируемые веб-страницы.

Используется React Bootstrap – это библиотека компонентов пользовательского интерфейса для React, которая использует компоненты Bootstrap, чтобы создавать дизайн и функциональность для приложений. React Bootstrap позволяет создавать адаптивный интерфейс для любых устройств, упрощает работу с CSS, и предоставляет удобный способ создания пользовательских компонентов.

Для обработки событий пользовательского взаимодействия и отправки запросов на сервер используется библиотека Axios. Библиотека позволяет делать запросы с различными параметрами, такими как заголовки, параметры запроса и тело запроса. Axios также умеет обрабатывать ошибки и возвращать данные в различных форматах, таких как JSON и XML. Это позволяет взаимодействовать с сервером более эффективно и удобно.

Библиотека React-router-dom используется для создания маршрутов веб-приложения и управления навигацией между страницами. Библиотека позволяет создавать ссылки, переходить на другие страницы, обрабатывать параметры запросов и отображать разные компоненты в зависимости от URL-адреса. Подобный подход позволяет создавать более сложные интерфейсы и улучшает навигацию в приложении.

Библиотека React-query используется для управления состоянием приложения и взаимодействия с сервером. Она позволяет делать запросы на сервер, кэшировать данные, обрабатывать ошибки и обновлять компоненты в зависимости от изменения данных. Это значительно упрощает разработку приложений и улучшает их производительность.

2.4 Блок взаимодействия пользовательского интерфейса с классами-контроллерами запросов

В приложении клиентская часть взаимодействует с серверной частью через REST API, который предоставляет классы-контроллеры приложения, то есть два класса, которые обрабатывают запросы клиентской части.

При отправке запроса с клиента на сервер, запрос сначала проходит через маршрутизатор в React-приложении, который определяет адрес и метод запроса. Затем запрос отправляется на соответствующий адрес на сервере.

На серверной стороне запрос обрабатывается контроллером, который содержит логику обработки запроса и взаимодействия с базой данных. Контроллер принимает запрос, выполняет нужные действия, получает или отправляет данные в базу данных через репозиторий, и возвращает ответ клиенту в виде JSON объекта.

Клиентская часть приложения принимает ответ от сервера и отображает его на странице при помощи React-компонентов. Если необходимо отправить данные на сервер, то 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 классов-контроллеров для обработки запросов

Блок контроллеров в приложении – это классы, которые отвечают за обработку HTTP запросов от клиента и передачу данных в сервисный слой приложения. Контроллеры реализованы на языке Java с использованием библиотеки Spring. Всего представлено два таких класса: `MainPageController` и `AuthenticationController`.

В классе `AuthenticationController` реализованы следующие запросы:

1. Запрос для аутентификации пользователей, принимает POST-запрос на пути «`/auth/authenticate`». Контроллер проверяет корректность введенных пользователем данных и возвращает JWT токен в случае успешной аутентификации.
2. Запрос для входа и регистрации пользователей, POST- запрос на пути «`/auth/register`». Контроллер проверяет корректность введенных пользователем данных и возвращает JWT токен в случае успешной регистрации.

В классе `MainPageController` реализованы следующие запросы:

1. Запрос для всей информации о сотруднике, которая будет отображаться на главной странице для авторизированного пользователя, GET-запрос на пути «`/mainUserInfo`».
2. Запрос для всех заявлений, назначенных на авторизированного пользователя, которые будут отображаться на его странице, GET-запрос на пути «`/ls`».
3. Запрос для подтверждения заявления, которое можно будет совершить с главной страницы, от авторизированного пользователя, POST-запрос на пути «`/ls/{id}`».
4. Запрос для всех событий, назначенных на авторизированного пользователя и отображаемые его странице, GET-запрос на пути `/events`».
5. Запрос для всех заданий, назначенных на авторизированного пользователя и отображаемые на его странице, GET-запрос на пути «`/tasks`».
6. Запрос для просмотра общей информации о всех сотрудниках, GET-запрос на пути «`/employees`».
7. Запрос для создания событий для пользователя, POST-запрос на пути «`/event`».
8. Запрос для создания назначения событий для пользователя на других пользователей, POST -запрос на пути «`/noticeevent/{idevent}`».

9. Запрос для создания заданий для пользователя и назначения их на других пользователей, POST-запрос на пути «`/task`».

10. Запрос для авторизированного пользователя на смену своего пароля, POST-запрос на пути «`/password`».

11. Запрос для авторизированного пользователя на создание заявления, POST-запрос на пути «`/lscreate`».

12. Запрос для авторизированного пользователя с правами администратора на редактирование данных, POST-запрос на пути «`/editemployee`».

Формат всех передаваемых данных будет JSON.

2.6 Блок бизнес-логики

Блок бизнес-логики обрабатывает запросы, отправленные клиентом. Серверная часть использует библиотеки Spring, и представляет собой ряд возможностей:

1. Регистрацию и авторизацию пользователей: при регистрации нового пользователя производится проверка на уникальность логина и пароля. При авторизации пользователей проверяются истинность логина и пароля.

2. Администрирование аккаунта только пользователем с правами доступа `ADMIN`. Оно включает в себе манипуляцию данными сотрудников, а также возможность блокировать их аккаунты.

3. Создание заданий и событий: пользователи могут создавать задачи и события для других пользователей. При этом система отображает их на страницах определенных сотрудников, на которых они были назначены.

4. Управление документами: пользователи могут создавать и хранить документы в системе, а также имеют возможность отправлять их прямиком своим начальникам, а они в свою очередь могут их подтверждать или опровергать. Это поможет облегчить взаимодействие между сотрудниками.

5. Организация доступа: система будет предоставлять три уровня доступа – администратор, пользователь и заблокированный. Каждый уровень будет иметь свои права доступа к определенным функциям приложения.

6. Оповещения: пользователи будут получать оповещения о новых заданиях, просьбах, событиях и других событиях в системе в виде отображения на личной странице.

7. Фильтрация данных: система будет предоставлять функции фильтрации данных в списках сотрудников.

8. Работа с сотрудниками: приложение будет предоставлять возможность создания профиля для каждого сотрудника, хранения и управления персональной информацией, а также управления их задачами, событиями и заявлениями.

9. Возможность для каждого аккаунта самостоятельно сменить пароль.

10. 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 используется Spring Security, который предоставляет нам множество инструментов для работы с безопасностью в приложениях.

В веб-приложении используется механизм авторизации на основе токенов JWT. Для этого настраивается специальный класс-конфигурация, в котором определяются правила доступа и их конфигурации. Также настроен процесс аутентификации, при котором проверяется истинность логина и пароля.

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. Такой подход удобен тем, что так как можно настроить токены так, что по истечению, например, некоторого времени пользователю будет необходимо пройти авторизацию снова.

При каждом запросе на сервер проверяется наличие токена в заголовке запроса и его валидность. Для этого создается и настраивается фильтр, называемый `doFilterInternal`, который проверяет токен: если он действителен, то пользователю разрешается доступ к запрашиваемому ресурсу.

Фильтр `securityFilterChain`, как и предыдущий, работает при каждом запросе и отвечает за конфигурацию безопасности, а именно за настройку Spring Security. Фильтр определяет правила безопасности для приложения и обрабатывает каждый запрос, проверяя, имеет ли пользователь доступ к запрашиваемому ресурсу. Если у пользователя нет прав доступа, то он будет перенаправлен на страницу аутентификации или получит ошибку доступа.

Таким образом, блок авторизации пользователей в веб-приложении с использованием Spring Security и JWT токенов позволяет обеспечить безопасность доступа к системе и защищать 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 применяется 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. Блок представляет собой пользователя с расширенными возможностями, а именно:

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 сотрудников: администратор сможет просматривать более углубленную информацию о сотрудниках.

2.10 Блок BLOCK

Блок представляет собой заблокированного пользователя и в данном приложении предназначен для ограничения доступа к функционалу и работе с данными.

Подобный блок необходим по причине политики безопасности, чтобы пользователи, которые по любой возможной причине, не имеют прав доступа к данным не смогли ими не только всячески манипулировать, а также не могли их получить и видеть в целом.

Блок заблокированных пользователей обеспечивает дополнительный уровень управления доступом, дополнительный уровень безопасности данных и снижает риск случайных или злонамеренных изменений информации.