### Описание архитектуры веб-приложения To-Do List для ведения списка задач

Архитектура программного обеспечения — это фундаментальная структура системы, которая определяет, как компоненты приложения взаимодействуют друг с другом и как они организованы для достижения поставленных целей. Хорошо спроектированная архитектура обеспечивает масштабируемость, поддерживаемость и надежность приложения. В данном приложении для управления задачами и проектами используется классическая трехслойная архитектура, которая разделяет приложение на три основных слоя: слой представления, слой бизнес-логики и слой доступа к данным. Каждый из этих слоев выполняет определенные функции и взаимодействует с другими слоями через четко определенные интерфейсы.

**1. Слой представления (Presentation Layer)**

**Описание**

Слой представления отвечает за взаимодействие с пользователем. Это внешний слой приложения, который обеспечивает визуализацию данных и обработку пользовательских запросов. В данном приложении слой представления реализован с использованием **Thymeleaf** — современного шаблонизатора для Java, который позволяет создавать динамические HTML-страницы.

**Основные функции:**

* **Отображение данных**: Слой представления получает данные от слоя бизнес-логики и отображает их в виде HTML-страниц. Например, список задач отображается в виде таблицы на странице.
* **Обработка пользовательского ввода**: Пользователь может взаимодействовать с приложением через формы (например, создание новой задачи или редактирование существующей). Слой представления передает эти данные в слой бизнес-логики для дальнейшей обработки.
* **Рендеринг страниц**: Thymeleaf используется для генерации HTML-страниц на основе шаблонов и данных, полученных от контроллеров.

**Пример работы:**

* Когда пользователь открывает страницу списка задач, контроллер (часть слоя бизнес-логики) получает данные из базы данных через слой доступа к данным и передает их в шаблон Thymeleaf. Thymeleaf рендерит HTML-страницу, которая отображается в браузере пользователя.
* Когда пользователь заполняет форму создания задачи и нажимает кнопку "Создать", данные формы отправляются на сервер, где они обрабатываются контроллером и сохраняются в базе данных.

**Преимущества использования Thymeleaf:**

* **Интеграция с Spring**: Thymeleaf легко интегрируется с Spring Boot, что упрощает разработку веб-приложений.
* **Динамические шаблоны**: Thymeleaf позволяет создавать динамические HTML-страницы, которые могут изменяться в зависимости от данных, полученных от сервера.
* **Поддержка современных стандартов**: Thymeleaf поддерживает HTML5 и современные веб-стандарты, что делает его удобным для разработки современных веб-приложений.

**2. Слой бизнес-логики (Business Logic Layer)**

**Описание**

Слой бизнес-логики отвечает за выполнение основных операций приложения, таких как управление задачами, аутентификация пользователей и обработка данных. Этот слой реализует ключевые функции приложения и обеспечивает связь между слоем представления и слоем доступа к данным.

**Основные функции:**

* **Обработка запросов**: Контроллеры (часть слоя бизнес-логики) принимают HTTP-запросы от слоя представления, обрабатывают их и возвращают результаты. Например, контроллер задач обрабатывает запросы на создание, редактирование и удаление задач.
* **Валидация данных**: Слой бизнес-логики проверяет корректность данных, полученных от пользователя (например, проверяет, что заголовок задачи не пустой).
* **Управление транзакциями**: Слой бизнес-логики управляет транзакциями при работе с базой данных, обеспечивая целостность данных.
* **Аутентификация и авторизация**: В этом слое реализована логика для аутентификации пользователей и проверки их прав доступа к определенным функциям приложения.

**Пример работы:**

* Когда пользователь создает новую задачу, контроллер задач получает данные из формы, проверяет их на корректность и передает в сервис задач. Сервис задач сохраняет данные в базе данных через слой доступа к данным.
* Когда пользователь запрашивает список задач, контроллер задач обращается к сервису задач, который, в свою очередь, получает данные из базы данных через репозиторий и возвращает их контроллеру. Контроллер передает данные в слой представления для отображения.

**Преимущества:**

* **Модульность**: Слой бизнес-логики разделен на контроллеры и сервисы, что упрощает поддержку и расширение приложения.
* **Гибкость**: Логика приложения может быть легко изменена или расширена без необходимости изменять другие слои.
* **Безопасность**: В этом слое реализованы механизмы аутентификации и авторизации, что обеспечивает безопасность приложения.

**3. Слой доступа к данным (Data Access Layer)**

**Описание**

Слой доступа к данным отвечает за взаимодействие с базой данных. Он обеспечивает выполнение операций CRUD (Create, Read, Update, Delete) и управляет хранением и извлечением данных. В данном приложении слой доступа к данным реализован с использованием **Spring Data JPA** и **Hibernate**.

**Основные функции:**

* **Работа с базой данных**: Слой доступа к данным выполняет запросы к базе данных для получения, сохранения, обновления и удаления данных. Например, репозиторий задач может извлекать список задач для определенного пользователя.
* **Маппинг объектов**: Hibernate используется для маппинга объектов Java на таблицы базы данных. Например, класс Task мапится на таблицу tasks в базе данных.
* **Управление транзакциями**: Слой доступа к данным обеспечивает целостность данных, управляя транзакциями при выполнении операций с базой данных.

**Пример работы:**

* Когда сервис задач запрашивает список задач для текущего пользователя, репозиторий задач выполняет SQL-запрос к базе данных и возвращает результат в виде списка объектов Task.
* Когда пользователь создает новую задачу, сервис задач передает объект Task в репозиторий, который сохраняет его в базе данных.

**Преимущества использования Spring Data JPA и Hibernate:**

* **Упрощение работы с базой данных**: Spring Data JPA предоставляет удобные интерфейсы для работы с базой данных, что сокращает объем кода, необходимого для выполнения операций CRUD.
* **Автоматизация запросов**: Spring Data JPA позволяет создавать запросы к базе данных на основе имен методов, что упрощает разработку.
* **Поддержка транзакций**: Hibernate обеспечивает управление транзакциями, что гарантирует целостность данных.

Трехслойная архитектура, используемая в данном приложении, обеспечивает четкое разделение ответственности между компонентами системы. Каждый слой выполняет свои функции и взаимодействует с другими слоями через четко определенные интерфейсы. Это делает приложение более модульным, поддерживаемым и масштабируемым.

* **Слой представления** отвечает за взаимодействие с пользователем и отображение данных.
* **Слой бизнес-логики** реализует ключевые функции приложения и обеспечивает связь между слоем представления и слоем доступа к данным.
* **Слой доступа к данным** управляет хранением и извлечением данных из базы данных.

Такой подход к архитектуре позволяет легко вносить изменения в приложение, добавлять новые функции и тестировать отдельные компоненты системы. Это особенно важно для веб-приложений, которые должны быть гибкими и адаптивными к изменениям требований.