Dropwizard- LeanIX: ToDoApp

The project outlines the development and implementation of a comprehensive task management system, designed to streamline and enhance productivity for users. The accompanying diagrams provide some visualization of the system's architecture, database, and workflow, illustrating the project's thoughtful integration of functionality and design to meet the users' needs effectively. I tried to put to think of a few things I couldn’t implement but I enjoyed playing around with it.

**User Flow:**

In the initial phase of our project, we encountered several challenges related to the integration and configuration of the Dropwizard framework, specifically with version 4.x and its associated dependencies. The intricacies of configuring the framework to suit our application requirements demanded a significant amount of time and effort, particularly in the following areas:

**Configuration Issues:** We faced difficulties with the config.yml file, which is crucial for setting up the Dropwizard application. Properly structuring this file to meet our specific needs proved to be a complex task.

**Database Integration:** The initial choice to use an H2 database (file-based) for testing purposes introduced complications. Despite the lightweight and easy-to-configure nature of H2, we encountered persistent issues in testing data persistence and retrieval, significantly impacting our development progress.

**File System Permissions and Paths:** We also grappled with challenges related to file system permissions and the specification of absolute paths within the application configuration. These issues affected our ability to smoothly manage application resources and data storage.

**External Plugins:** Integrating external plugins required significant effort to ensure compatibility with the version of Dropwizard being used, as well as adherence to its design patterns.Integrating external plugins required significant effort to ensure compatibility with the version of Dropwizard being used, as well as adherence to its design patterns.

Given the constraints and the time invested in resolving these issues, a strategic pivot was deemed necessary to maintain project momentum. This led to the decision to transition from using an H2 database to adopting PostgreSQL. Furthermore, to streamline development, testing, and deployment processes, I opted to containerize our application using Docker. This approach not only addresses the initial challenges but also enhances our application's portability and ease of deployment.

**Bigger Picture :** Cartoon of people looking at a list of things

Description automatically generated

Okay lets try ….

C

O

N

F

I

G

Front End

**Mr. Postman**

**Database**

PostgreSQL/H2(Embedded)

**Persistence Layer**

*Todo Repository and SubTask Repository*

Storage Logic

**Controller**

**TodoResource and SubTaskResourc**

Interface to business logic

**Service(Business) Layer**

?????????????????????

Calendar and Calculations

**Domain Layer**

*User, SubTask and Todo*

Authentication, Json Translation.

ToDoApp **Dropwizard**