Design Documentation

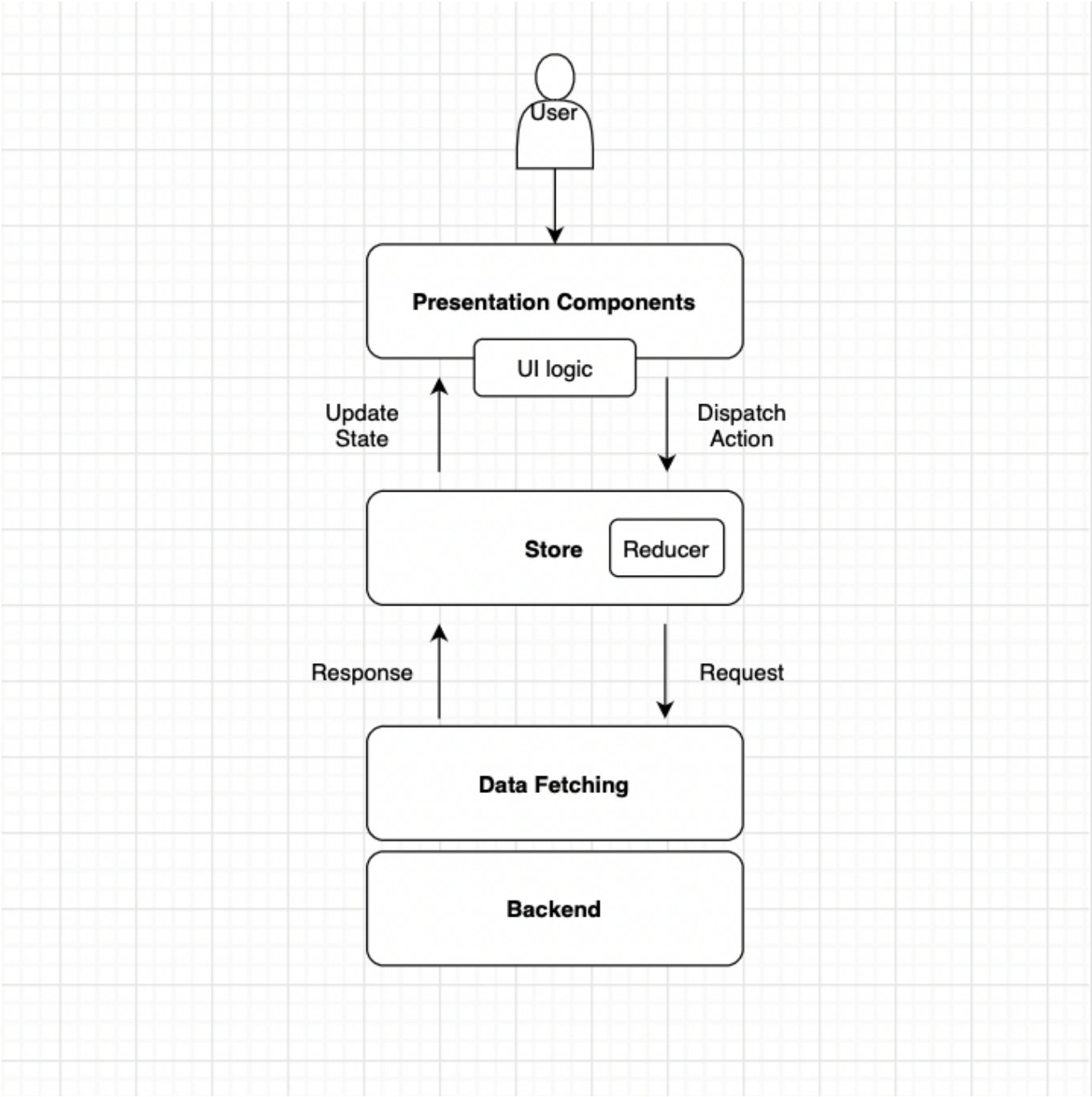
You can find the current Design-Documentation in our github wiki here:

https://github.com/amosproj/amos2023ss01-apache-pulsar-ui/wiki/Design-Documentation

To meet the requirements of the sprint, we also transformed the state of the wiki from 30.05.2023 to this pdf.

Frontend

The following diagram shows the architecture of the Frontend of our application.



Store

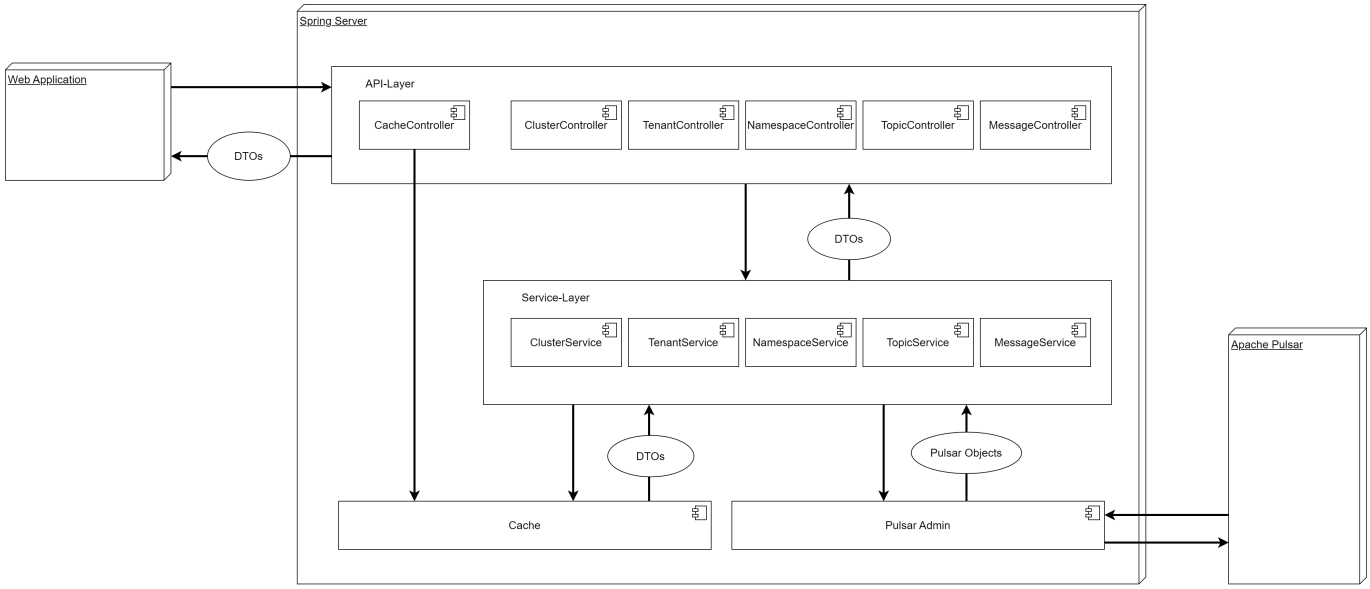
Under the hood, we use Redux for state management. It means that we have a single store for the whole app and every part of it can access (and modify) any data. That provides us with a single source of truth which is one of the three fundamental principles of Redux.

Reducer

Reducers are functions that describe how the state should be modified as a result of the action. There is just one reducer associated with the store but multiple reducers can be combined into one.

Backend

The following diagram shows the architecture of the backend of our application.



Controller

The controller classes provide REST API endpoints for the frontend to send or receive data and in turn use different services to interact with Pulsar.

Our controllers are:

- ClusterController: Provides a list of all clusters in the instance with aggregated information about each cluster like the amount of namespaces etc.
- TenantController: Provides a list of all tenants across all clusters.
- NamespaceController: Provides a list of all namespaces across all tenants.
- TopicController:
    - Provides a list of all topics across all namespaces.
    - Provides a list of all topics of a given namespace and tenant.
    - Provides a topic by name.
    - Allows to create a new topic.
- MessageController:
    - Provides a list of all messages of a given topic and subscription.
    - Allows to send a message.
- CacheController: Allows flushing the cache.

Services

The services are an abstraction called by the controllers to provide them with the requested information of the Pulsar instance. They use the Pulsar Admin Java API to communicate with the Pulsar instance.

DTOs

We map the information returned by the Pulsar Admin Java API to custom DTOs to structure and filter the information in a way that makes it easier for the Frontend to process. Since we don't have a lot of domain logic in the services, we also use them in the service-layer to avoid duplication.