ANGULAR NGRX PWA APP

**OVERVIEW**

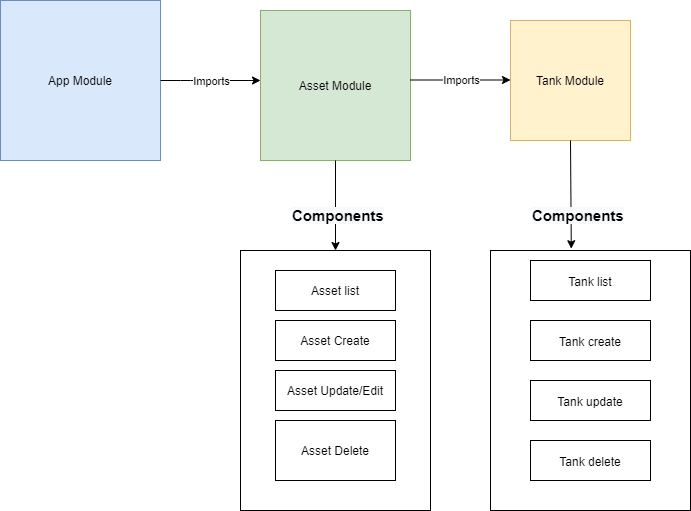
Fortune is a simple asset-tank management system built on Angular. This facilitates users to add, update, modify and delete assets and tanks based on their choices.

This application uses angular for frontend, ngrx for state management and C# webapi.

**1.1 Block Diagram**

As the following figure illustrates, our application will consist of two primary modules, namely App, Assets and Tanks. The Asset module, in turn, will have four custom components, namely Assets List , Asset Add , Asset Edit, Asset Trash.

Similarly for Tank Module, Tanks List , Tank Add , Tank Edit, Tank Trash.







**In general, the application interacts with a C# API to perform CRUD operations on data.**

Therefore, implemented a simple C# API with Mongo Collections as Lightweight Database that exposes the below endpoints. We will use this API to connect from the Angular application and carry out data operations.

**1.2 ASSETS**

\*\* Retrieve All Assets\*\*

GET https://localhost:5001/assets

\*\*Create an asset\*\*

POST https://localhost:5001/assets

\*\*Update an asset\*\*

PUT https://localhost:5001/assets/{assetId}

\*\*Delete an asset\*\*

DELETE https://localhost:5001/assets/{assetID}

**STATE MANAGEMENT**

We are using NGRX to build the reactive application. It facilitates:-

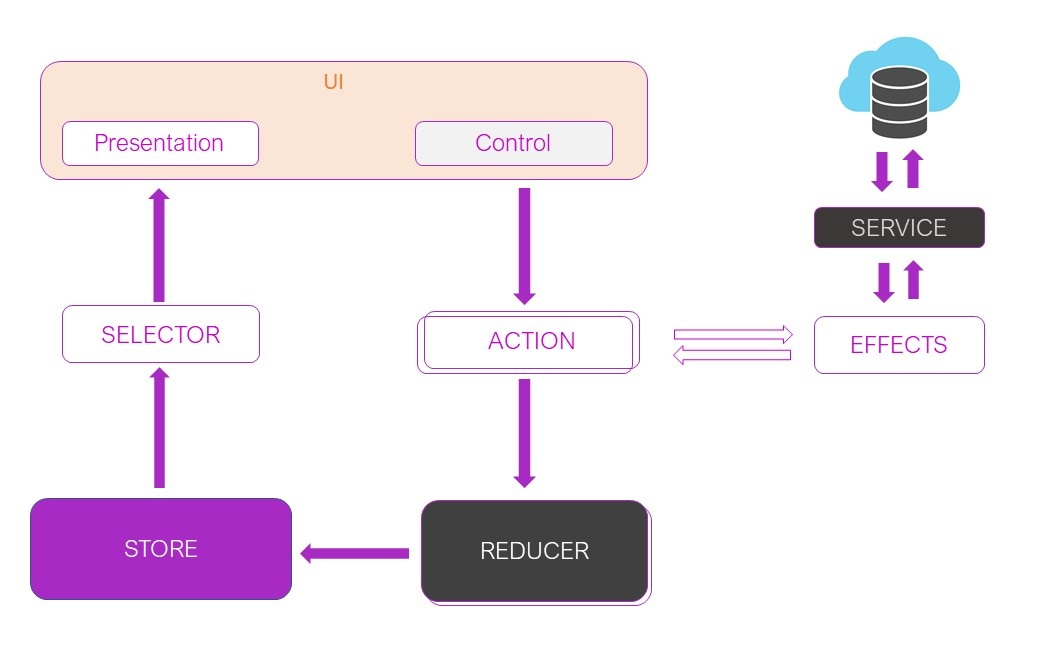
* Managing global and local states.
* Isolation of side effects to promote a cleaner component architecture
* Entity Collection management
* Developer tooling that enhances developer experience when building many types of different applications.

This application includes following libraries for state management:

1. ngRx/store
2. ngRx/effects
3. ngRx/entity
4. ngRx/schematics
5. ngRx/store-devtools

Let’s look at the NgRx implementation — there are several components to understand.

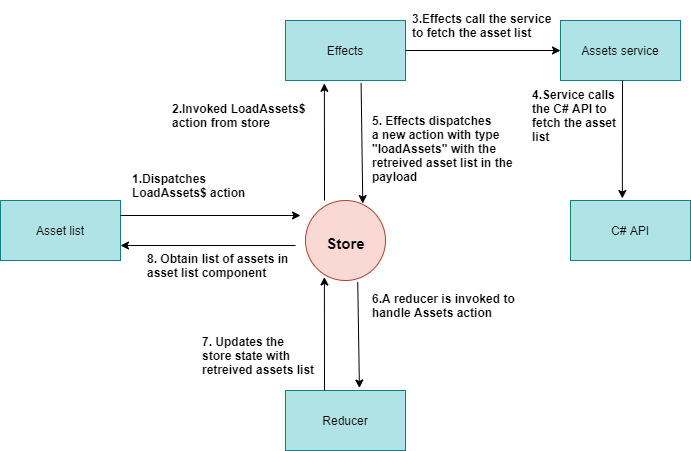
* **Store**: The store is what holds the app's state.
* **Action**: A unique event dispatched from components and services that describe how the state should be changed. For example, ‘Add Customer’ can be an action that will change the state (i.e. add a new customer to the list).
* **Reducer**: All the state changes happen inside the reducer; it responds to the action and, based on that action, it will create a new immutable state and return it to the store.
* **Selector**: Selector is a function used for obtaining a part of the state from the store.
* **Effect**: A mechanism that listens for dispatched actions in an observable stream, processes the server response, and returns new actions either immediately or asynchronously to the reducer to change the state. Please note that we are not using 'effect' in this example app.



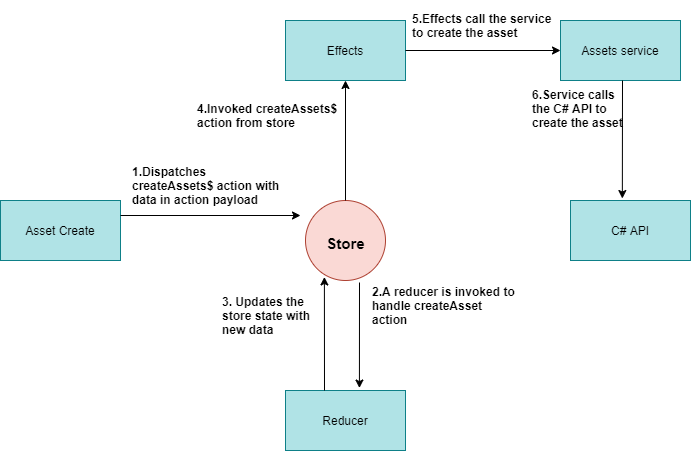
Lets us understand state management more precisely with state management sequence and activity diagrams for different operations

**1.3 State-activity Diagram**

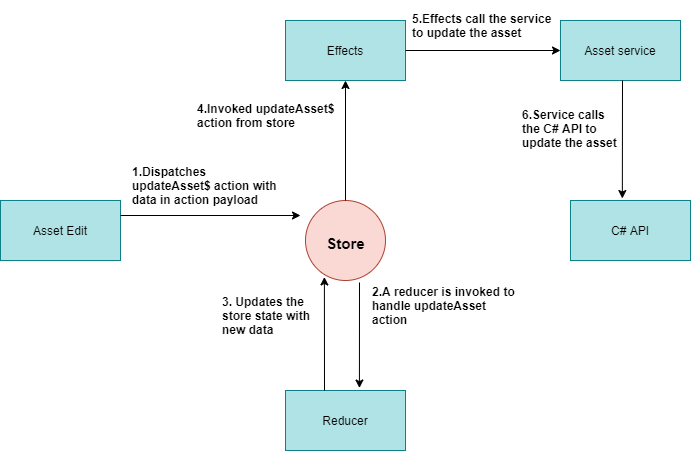
**1.3.1 Retrieving the asset list**



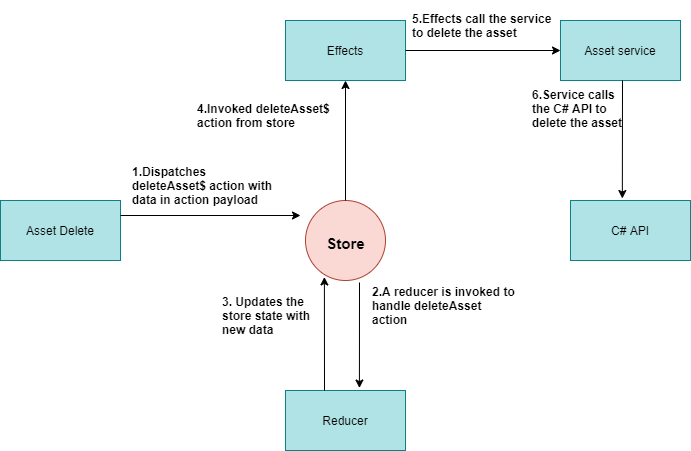
**1.3.2 Creating an asset**



**1.3.3 Updating an asset**



**1.3.4. Delete an asset**



**1.4 TANKS**

\*\* Retrieve All Assets\*\*

GET https://localhost:5001/tanks

\*\*Create an asset\*\*

POST https://localhost:5001/tanks

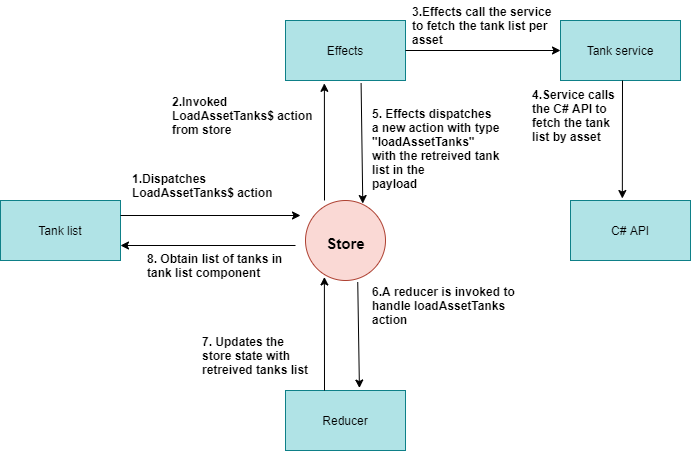
\*\*Update an asset\*\*

PUT https://localhost:5001/tanks/{tankId}

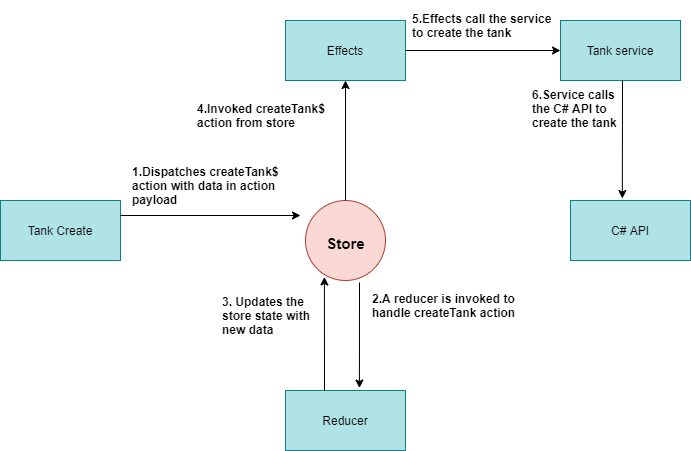
\*\*Delete an asset\*\*

DELETE https://localhost:5001/tanks/{tankId}

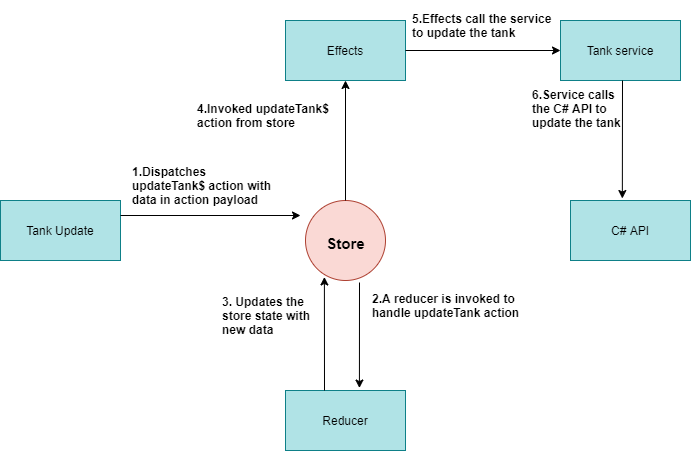
**1.4.1. Retrieving Tank List**

****

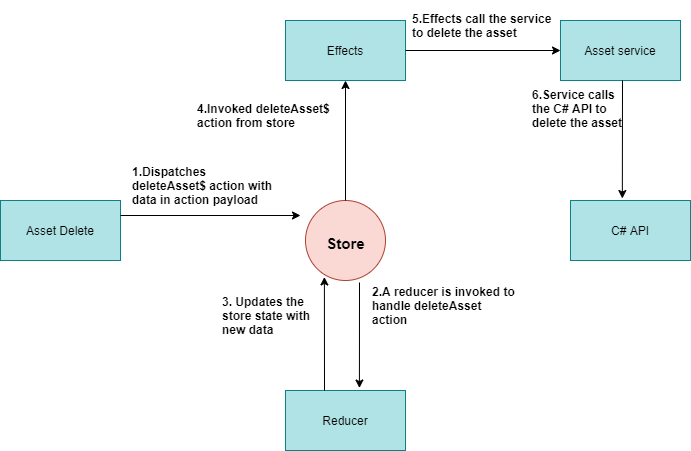
**1.4.2.Creating a tank**

****

**1.4.3. Updating a tank**

****

**1.4.4. Delete a Tank**

****

**2.1 Setting up PWA**

### **What is PWA?**

A progressive web app offers a high level of user experience because it has the same features as native apps have. It offers following features:-

1. Progressive
2. Responsive
3. Connectivity Independent
4. App-like style navigation and action
5. Up-to-date
6. Safe
7. Installable

To make our application PWA, all you need to do is run a single command:

ng add @angular/pwa

This command will install all of the required pwa dependencies to the project. It adds the following requirements to the project.

* Create an icons folder into src/assets directory
* Create a [service worker](https://web.dev/precaching-with-the-angular-service-worker) with a default caching configuration.
* Create a [manifest file](https://web.dev/add-manifest), which tells the browser how your app should behave when installed on the user's device.
* Add the [theme-color](https://web.dev/themed-omnibox) <meta> tag to index.html.
* Add a link to the manifest file in index.html.

**NGX-DEVICE-DETECTOR SERVICE**

This service is used to detect the device-type i.e, mobile, tablet or desktop. We are using a ***device-detector.service.ts*** file. Following steps to be taken while configuring device-detector:-

1. npm install ngx-device-detector.
2. Create a service file device-detector.ts which should be injectable across components.
3. Initializing constructor with a call to device-detector.service.ts
4. Using isMobile(), isTablet(), isDesktop() methods inheritance calls in the component.

**2.2. RUN THE APPLICATION**

Following steps are needed to run the application:-

Step1: Get inside the project folder and run.

npm install

This will install all the required dependencies in a node\_modules folder inside your application’s directory.

Step 2: Run following command.

ng serve

The above command will open the angular app on the following URL [http://127.0.0.1:](http://127.0.0.1:8080/)4200 and also give you the following URLs, you can check your app by entering one of the URL in the browser’s address bar