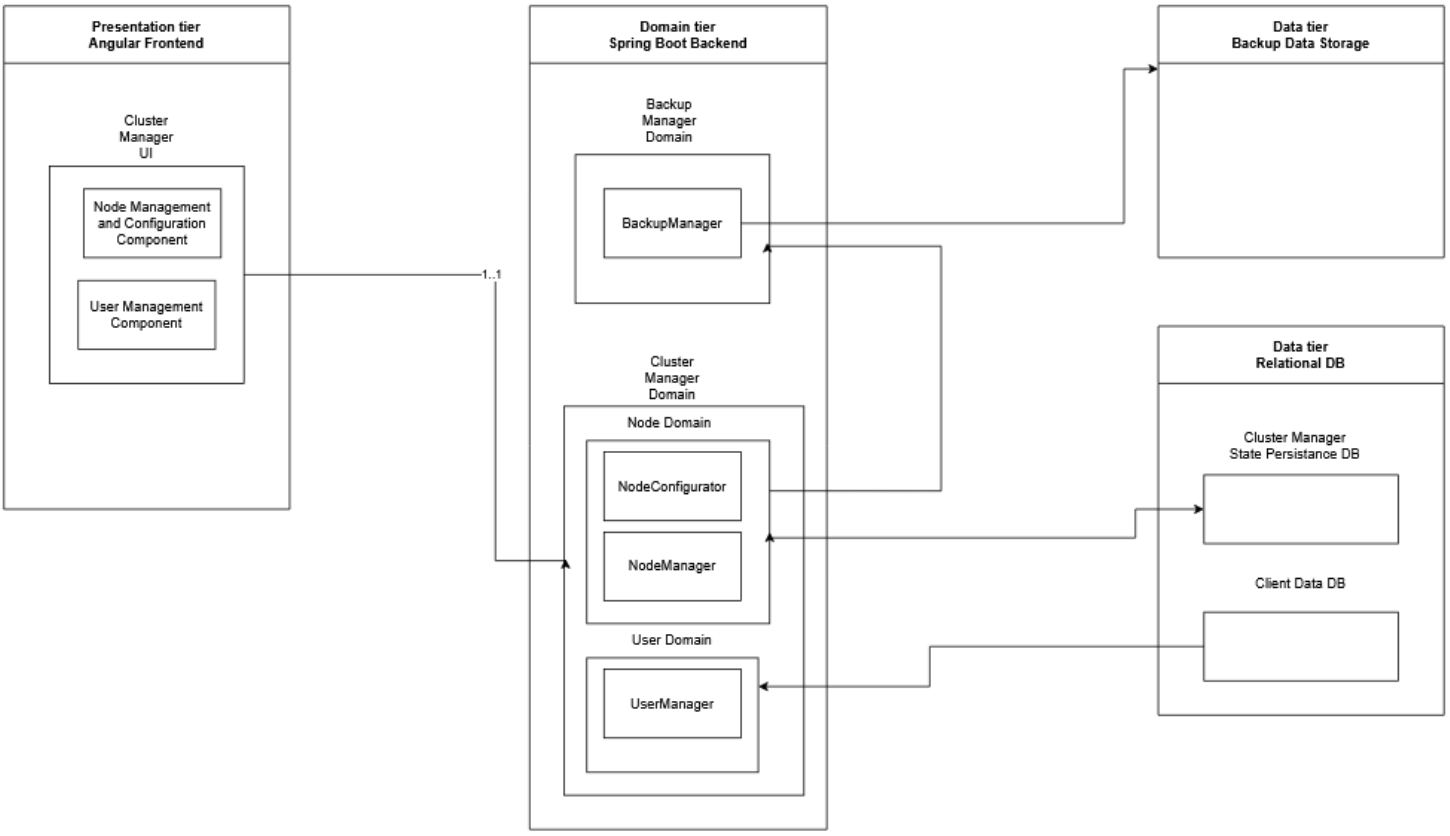
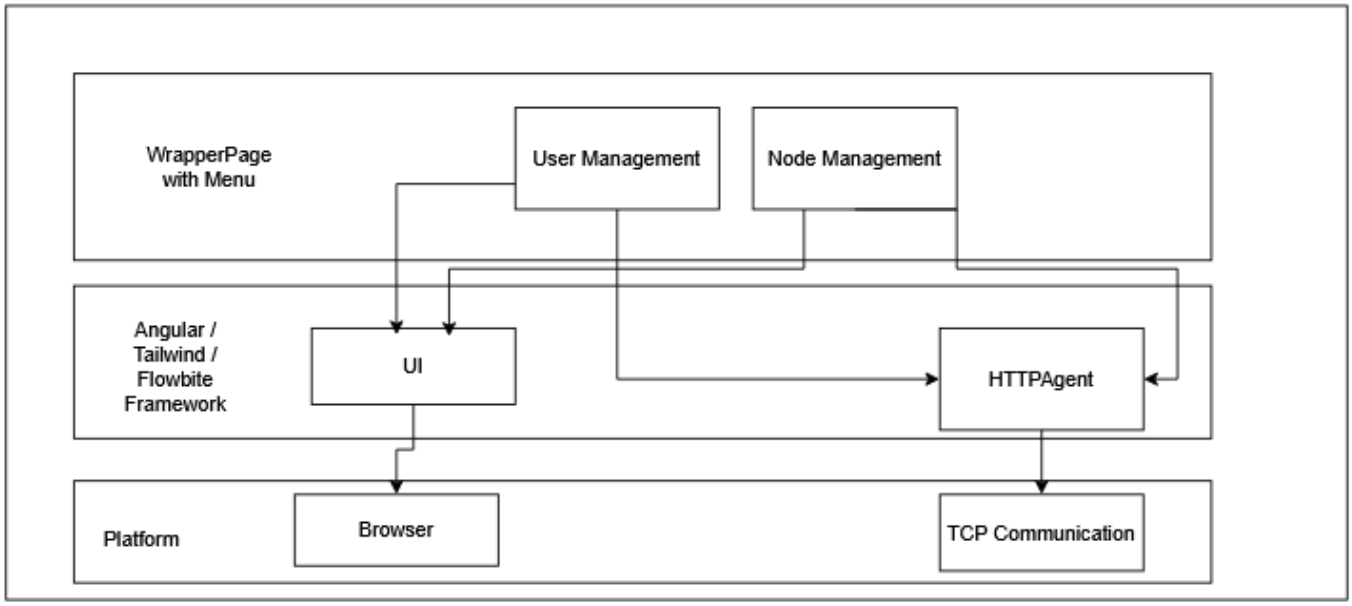
Runtime Architecture

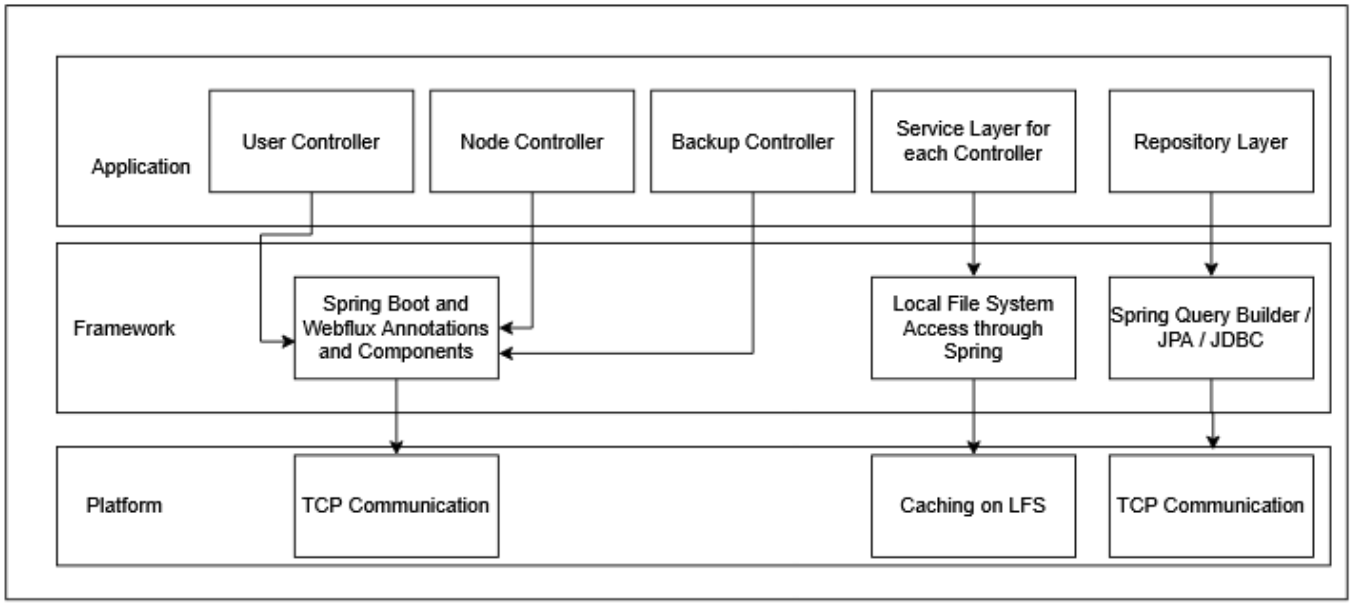


Code Structure

Frontend



Backend



Techstack

- **Frontend:**
  - **Linux**
  - **Docker**
  - **NodeJS + NPM**
  - **TypeScript**
  - **Angular, Tailwind, Flowbite**
- **Backend:**
  - **Linux**
  - **Docker**
  - **Java + Maven**
  - **Spring Boot, Webflux**
  - **(maybe Consensus Algorithms like ZooKeeper)**
- **Generally:**
  - **Docker compose**
  - **Git**
  - **GitHub Repository**
  - **GitHub Actions as CI**
  - **Relational DB (which exactly not clear yet, Postgres, MySql,..)**
  - **Caching DB (e.g. Redis, local file system)**

Description:

Runtime Architecture:

- **Docker Compose to simulate cluster**
- **Frontend as a Web Page with Angular and TypeScript deployed by a independent docker container**
- **Frontend represents a UI to manage users as well as manage (adding / removing nodes, configure each and view their current state)**
- **The frontend communicates with the current cluster manager API of the current cluster manager node**
- **In the cluster there can be nodes, where any node can be a backup manager, backup storage or cluster manager node. Only one cluster manager node can be present at the same time**
- **Multi-Role behavior is represented by a docker image that has all the backend code and API capabilities are limited or enhanced at runtime depending on the role (a cluster manager can still act as a backup manager and be responsible for backups while providing cluster manager services)**
- **The state of the cluster manager (the managed users, managed nodes & configurations) are stored in two databases from which a new cluster manager can restore its state**
- **Cluster managers are decided by configuration on startup or by a consensus algorithm (leader election, e.g. ZooKeeper)**

Additional Overview: