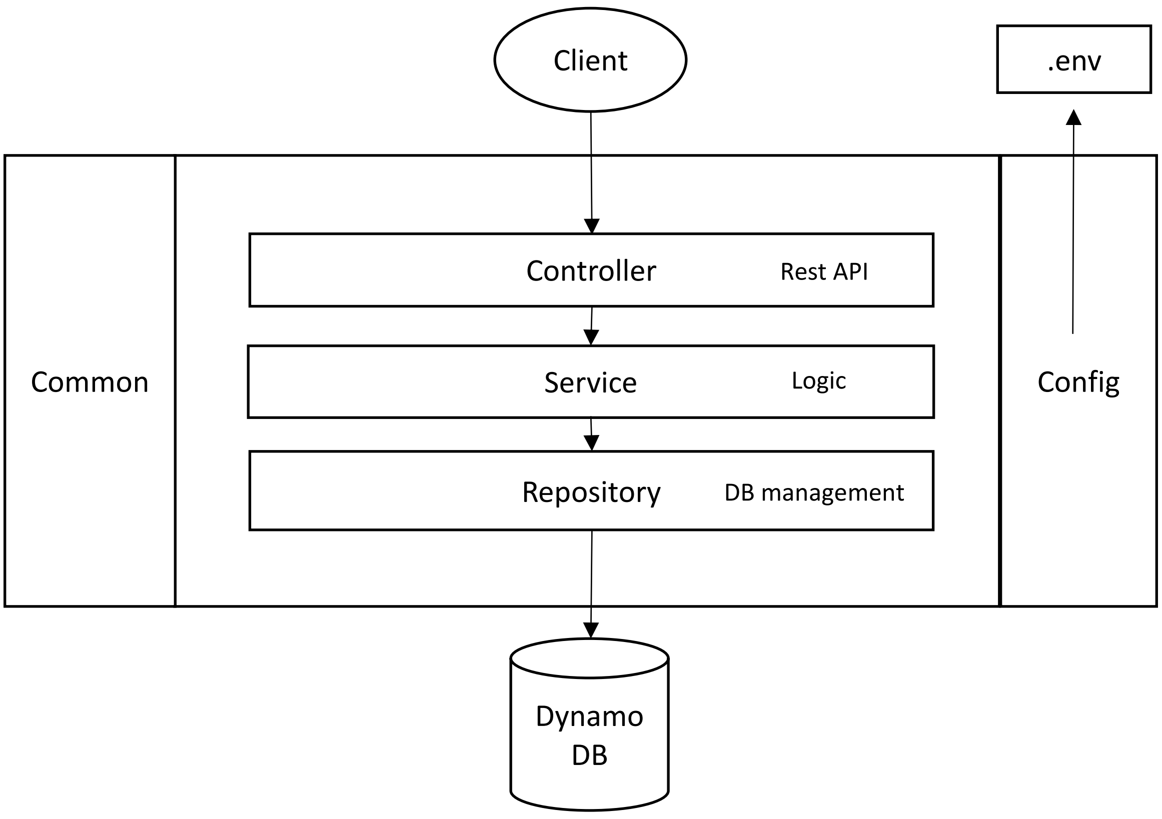
Lamoureux Adrien

2019/01/20

Calm-island - Assessment report

Architecture overview



The chosen architecture is a layered one. Every layer has his own responsibilities so it makes the code easier maintainable and readable.

The default configuration is in config.js. This configuration can be overridden through an .env file.

More information concerning the technologies can be found in the Readme file.

Logs of the tasks

Initially, the scheduled time for this assessment was about 2 days as followed:

- 1 day to setup the architecture and write the report
- 0.5 to sign-up
- 0.5 to sign-in / sign-out

However, the final time is about 2.5 days because:

- 1.5 days on the architecture setup alone. Basically, I spent more time than scheduled some elements like Jasmine integration and DynamoDB learning curve
- 0.25 days to cleanup, to write the documentation and to deliver the code
- 0.25 to sign-up because it's handy once the architecture is ready
- 0.5 to sign-in / sign-out because Passport.js helps a lot

Weaknesses

Using Node.js has several drawback that should seriously be considered to go further in the development.

Single-threaded environment

The Node.js server should be used as a microservice, for a dedicated set of tasks. Basically, the logic core of each Node.js server should be kept limited and avoid heavy computations. For example, in our case, the developed Node.js server should be used only for User Management.

Event-based

JavaScript is an asynchronous, event-based, language. Therefore, "Callback hell" is often a problems. Using Inversify and layered architecture is a pattern to avoid callback based architecture.

Exception management

Adding a middleware and extra layers to manage exceptions is missing in this project.

More tests

More tests need to be added, including:

- Tests cases for existing Unit tests like registration, …
- More integration tests with DynamoDB
- Write Unit tests and Integration tests for session management with multiple users, …

Production

There is several things to do because the architecture is not ready to go in production. I think these are the main ones.

Security

Security is one of the biggest concern when you go in production. Minimum would be to use HTTPS (SSL data encryption) and to encrypt stored users information (password…) in the DynamoDB.

Data optimizations

Because the production mode could involve a large numbers of queries, it’s necessary to provide an architecture to avoid deadlocks. For example, by using cacheable data and queries, using tools to measure memory and time, horizontal scalability with more microservices…

More configurations

Simply, define a setup for development, validation and production mode. Moreover, an administration tool should be available too for back-office services.

Documentation

Rest API, versioning and guidance should be provided before going to production. Therefore, everyone would be able to maintain (fix and improvement) the product quicker.