Assignment implementation

https://github.com/jarkko-turpeinen/assignment.git

REST services by JAX-WS on IBM Cloud Liberty server. Client is single page application based on React JS. Database on IBM Cloud IBM Cloudant.

Java implementation is straightforward without design pattern hell. Reasons are small size of an application and desire to keep codebase as simple as possible. Less code is less bugs and even trainee with low experience must be able to maintain codebase successfully.

Code is tested by JUnit tests and integration tests. Continuous Integration is done in Circle CI <http://circleci.com>.

Integration tests use Cloudant database named “integrationtests”. Application use database named “equipments”. Unit tests and integration test are separated by Gradle sourcesets.

User interface is bootsrapped by create-react-app and it is not ejected because no customizations were needed. There are only few React components implemented: App, EquipmentForm (container), DatePicker (stateful) and EquipmentList (stateless). I could not see any benefit to extract app further pieces.

Container component takes care of communicating to REST services. It uses *async await* technique to avoid unnecessary callbacks and keep code base more clean and readable (opinionated).

No frameworks (eg. Redux) are used for same reasons as Java is kept small and understandable. Short learning curve for developers without knowledge of fancy frameworks and patterns is desirable.

User interface validation is *thinking outside of the box* version just to showcase state handling and creativity. It’s also more fun to use, at least for me.

Builds for test and production and running in development mode is separated by *env.* Every environment got its own settings that are automatically used by npm run [start | test | build].

Ideas for improvement:

* Single validation codebase as JS. Browser and Java (by Nashorn) may use same validation code.
* Result listing by ready npm library that is widely used and handles paging and sorting and batch. loading.
* Login services App ID integration or similar is required in production.

Gateway API is not implemented but required in production. REST services should reside behind gateway that handles authorization.