Project description.

Front side web app that will use an API for signing documents.

In essence, the app will be able to authenticate users with federated identities, will be able to show a form to authenticated users that they can use to sign documents, and will be able to hand back the signed documents to the users.

The app will be developed in modern JS with React and Redux. This app will be responsive and internationalized, and fully tested and documented.

Done so far.

Frontend development infrastructure.

Built around Webpack and Babel.

The frontend code is not plain JS that a browser can understand. This is due to 2 different factors:

- It uses React's JSX and ES6 APIs, not understood by the browsers;
- It is modularized in the style of ES6.

Babel is a transpiler that can turn JSX and ES6 code into JS code understandabe by the browsers. Webpack is a bundler, that can gather all the modules used by the app and bundle them into a single JS file.

React infrastructure.

We use React to develop the front app in terms of composable components.

Docker environment.

We use Docker to deploy a local development environment, that allows us to test the integration of all the different pieces.

So far this environment just spawns a single container, with an NGINX process that serves the index page and the bundles built by webpack.

Testing infrastructure.

All the JS code will be unit tested.

We use Karma as a test runner. Jest seems more popular, but Karma allows us to run the tests in a real browser.

As testing framework, we use Mocha, with Chai as assertions library.

Also, we use React Testing Library to mount React components in the tests, and to deal with events and the lifecycles of the components.

Makefile.

We will use a Makefile to hold all the scripts needed for development and deployment.

Still to do.

The total estimated effort to complete the pending tasks are 26 developer days.

Redux infrastructure.

We will use Redux to keep all the state of the front app in a central store, and to communicate changes in the central state to the appropriate React components.

estimated effort: 3 d.

i18n infrastructure.

We will use react-intl to internationalize and localize the front app.

estimated effort: 2 d.

Authn mechanism.

When a user loads the front app, the app will check that there is an authn token (probably a JWT) available, and in case there isn't, it will direct the user to some authn service.

estimated effort: task still not clear enough to make estimations.

Main container, header and footer components.

The markup and style common to all pages in the app. This must be responsive.

estimated effort: 5 d.

Infrastructure for notifications to the user.

Whenever the user interacts with the app, it will need to inform the user about the result of the interaction.

estimated effort: 1 d.

"Fetching" infrastructure.

Whenever the app is waiting for some request, we need to disable controls to avoid further interactions from the user.

estimated effort: 1 d.

Splash screen.

A React component to be shown while the app loads.

estimated effort: 1 d.

Form infrastructure.

We will use redux-form to develop the needed forms.

estimated effort: 2 d.

PDF uploading form.

Once the user is authenticated, she will be shown a form from which she can upload PDFs to be signed.

estimated effort: 5 d.

Result page.

After uploading the PDF, the user will land in a page that allows her to get the signed PDF, and possibly access other documents previously signed.

estimated effort: 3 d.

Documentation.

For any future developer.

Estimated effort: 3 d.