UCLH PEACH Form Builder

Technical Guide

Overview

The document provides the summary and overview of the most information that has been collected during the development of UCLH PEACH Form Builder. The section covered include a general introduction to openEHR - a data standard in the hear of the system; EHRScape - an openEHR-oriented backend system that our team used; structure of our application, including the description of design patterns we employed; main technological stack used; references to useful resources.
A more detailed description of our findings is provided on the team 41 website.

openEHR

"openEHR is a virtual community working on means of turning health data from the physical form into electronic form and ensuring universal interoperability among all forms of electronic data. The primary focus of its endeavour is on electronic health records (EHR) and related systems."[1]

In essence, openEHR is a very comprehensive standard for how medical data needs to be structured. Aim of the specification is to provide schemas for all common types of medical data measurements. OpenEHR architecture consists of several core artefacts:

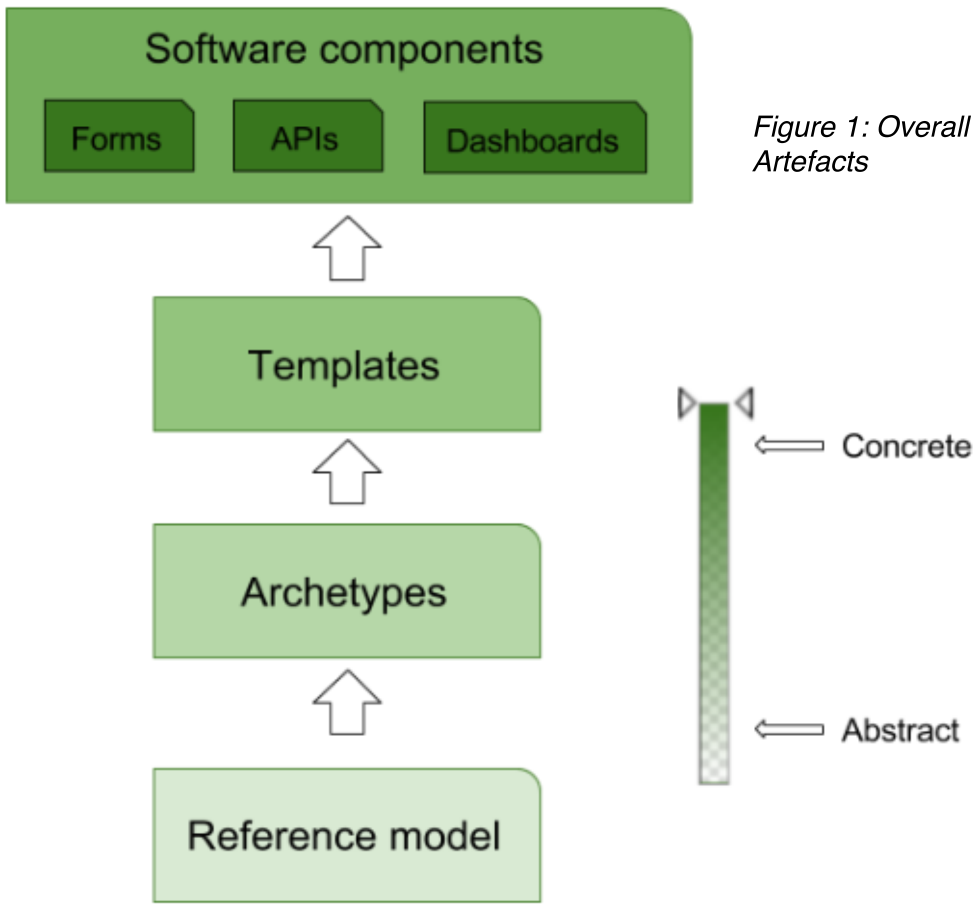


Figure 1: Overall Structure of openEHR Artefacts

Reference model

This is an underlying component that defines ways in which data obeying to openEHR standards is stored, created and processed. This artefact can be understood as an openEHR-specific information model, that determines all possible data. All data points, groups and sets in openEHR derive from reference model.

Archetypes

Archetype is a collection of data points that are related to a particular entity. If we regard reference model as a large collection of data points, than we could say that archetypes impose restrictions on the model, so that only relevant points are available. This collection of data point is called a data group. An example here would be “blood pressure” data group, that can have “systolic” and “diastolic” as data points. [1]

Templates

Both data points and groups are very generic and not related to any particular medical event or document. In order to implement this, event-specific data sets are created. They serve as a basis for software components adhering to openEHR standards. Each template is composed from a number of archetypes, but data points within each archetype can be further restricted so that the template provides only relevant information to a particular case. For instance, a template that includes “blood pressure” archetype is constructed. If we are only interested in systolic blood pressure, than “diastolic” data point can be restricted and it will not be displayed in a final software component. An example of entities for which templates can be created include “Health Risk Assessment form” and “GP encounter log”.[2]

Software components

Finally, templates can be used to generate software components that will be used in a medical system. The components will contain representation of data points that were included in the template. Possible components include input and output forms, APIs, dashboards.

EHRScape

EHRScape is a backend system that is specifically designed to work with openEHR based systems. Communication with the backend server is done through a set of asynchronous API calls.

Each system is assigned a separated EHRScape domain. The domain contains sets of EHRs - electronic health records. Each EHR represents a single person. Each EHR can store a number of compositions - essentially documents - associated with this particular person. Compositions are stored according to templates that have been previously saved in the particular EHRScape domain. So, in order to be able to save a particular type of data, a user must load a corresponding template into the system.

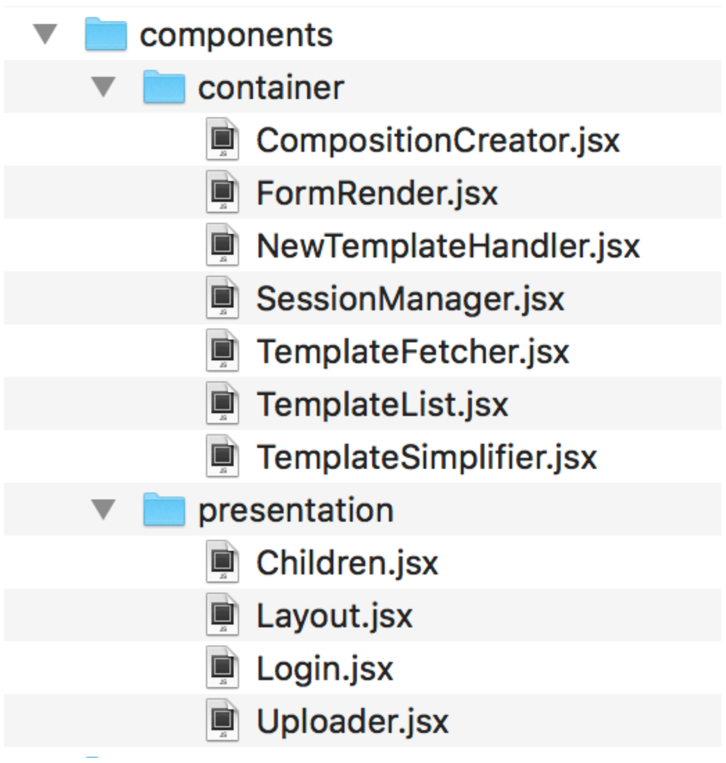
Application structure

Our system is based on ReactJS library, which enables developers to separate an application into a set of *components* - encapsulated objects, each in charge of a specific part of user interface or its underlying logic.

In the development of the system, it was convenient to split it into container components - the ones in charge of data processing and presentation components - ones that display it to a user. Hence, we employed a corresponding design pattern.

For instance, a Layout component, that determines overall layout of the application, can be taken as an example. As it can be seen on the Figure 2, there is a single presentation component associated with it - Layout.jsx.

However, as this element encompasses a a big number of children elements, it has a number of logical components - containers, that determine logic of the component. The containers associated with include NewTemplateHandler, SessionManager and Template list.



```
▼ components
  ▼ container
      CompositionCreator.jsx
      FormRender.jsx
      NewTemplateHandler.jsx
      SessionManager.jsx
      TemplateFetcher.jsx
      TemplateList.jsx
      TemplateSimplifier.jsx
  ▼ presentation
      Children.jsx
      Layout.jsx
      Login.jsx
      Uploader.jsx
```

Figure 2: Overall Structure of the Form Builder components

The following class diagram provides and overview of the core components of constituting the system:

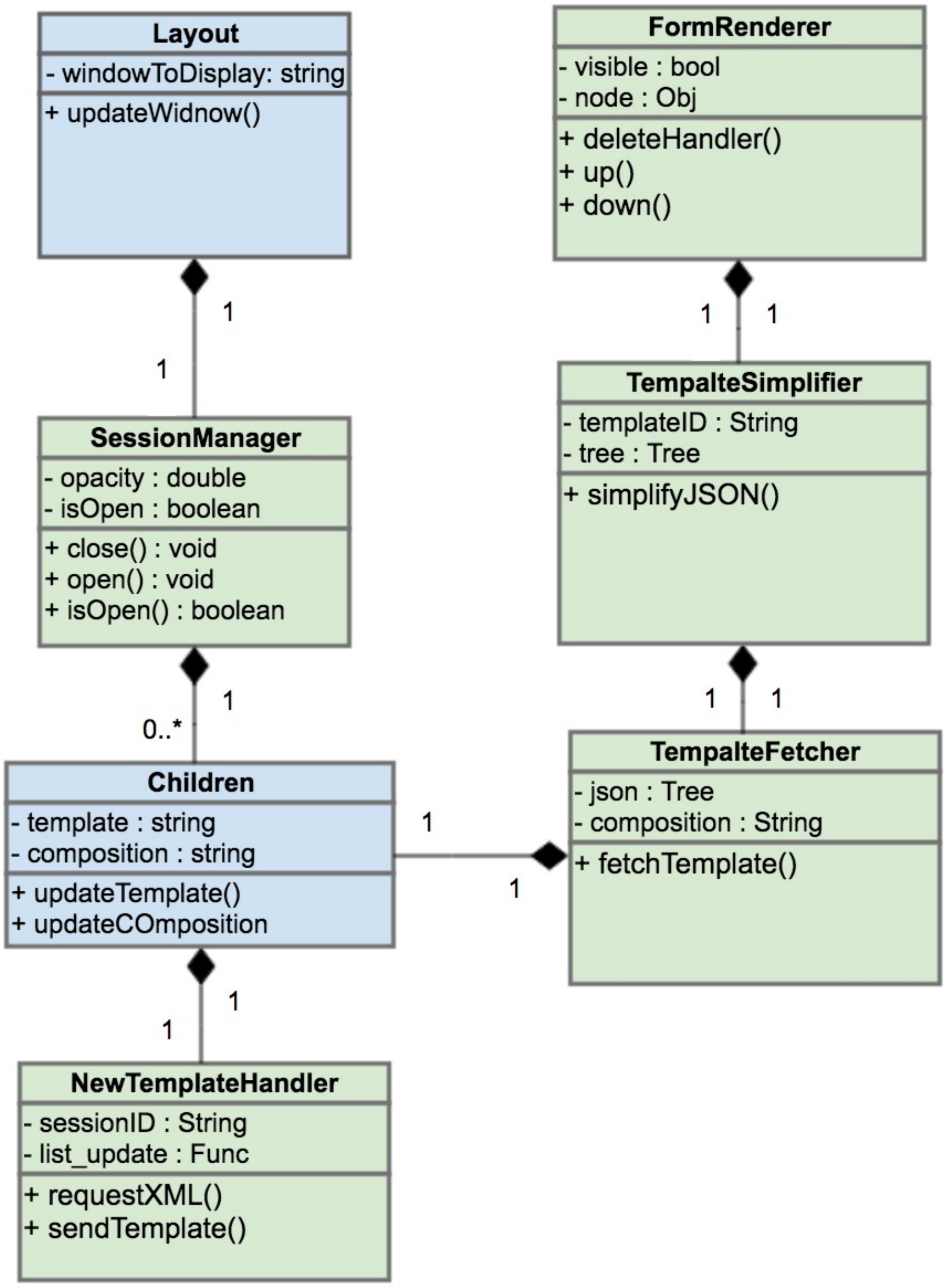
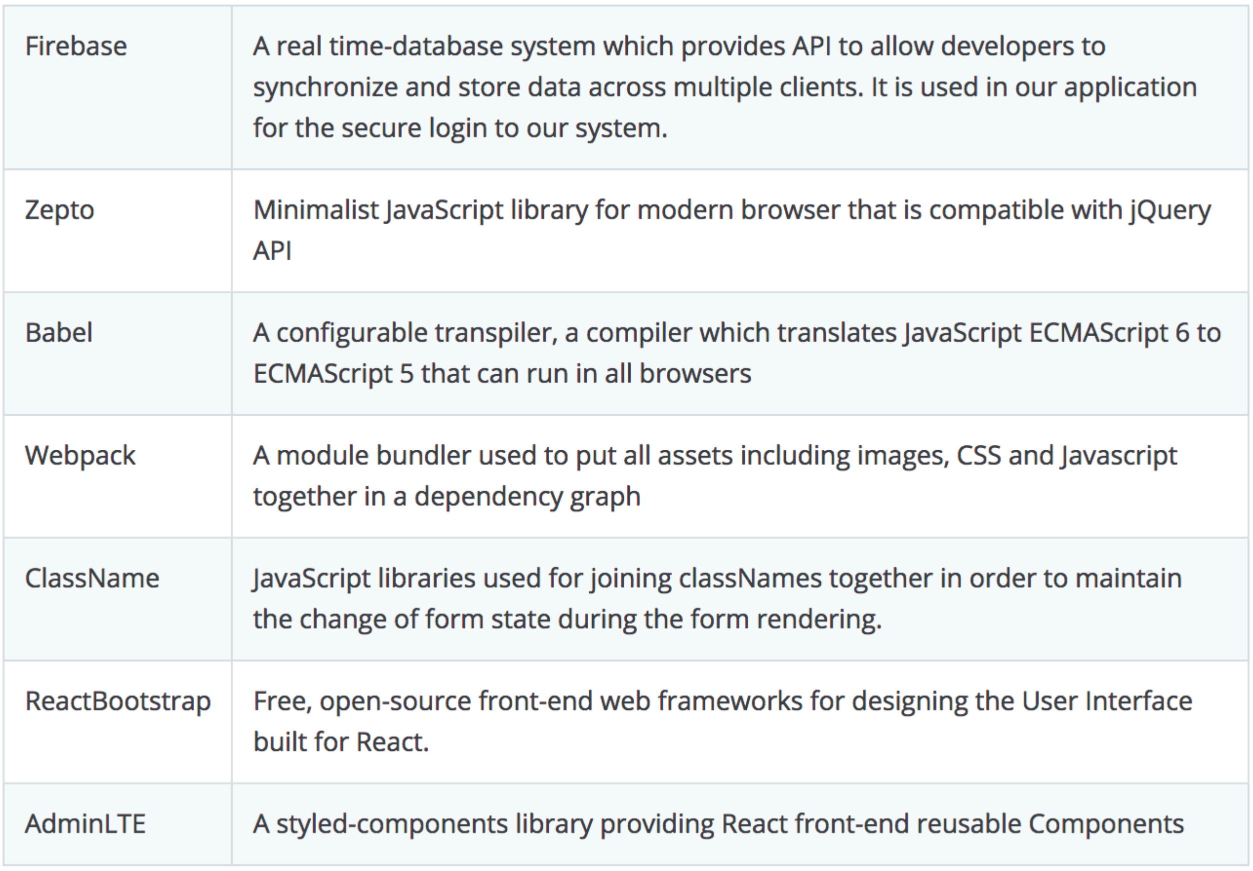


Figure 3: Form Builder Class Diagram

Main technological stack

This the list of core libraries that were used in the course of system development



Firebase	A real time-database system which provides API to allow developers to synchronize and store data across multiple clients. It is used in our application for the secure login to our system.
Zepto	Minimalist JavaScript library for modern browser that is compatible with jQuery API
Babel	A configurable transpiler, a compiler which translates JavaScript ECMAScript 6 to ECMAScript 5 that can run in all browsers
Webpack	A module bundler used to put all assets including images, CSS and Javascript together in a dependency graph
ClassName	JavaScript libraries used for joining classNames together in order to maintain the change of form state during the form rendering.
ReactBootstrap	Free, open-source front-end web frameworks for designing the User Interface built for React.
AdminLTE	A styled-components library providing React front-end reusable Components

References and useful resources:

[1] Heard, S. and Beale, T. (2016) What is openEHR? Available at: http://www.openehr.org/what_is_openehr (Accessed: 10 December 2016).
[2] Falkman, D. (2015) MVC Frameworks for Building PHP Web Applications. Available at: https://www.lynda.com/CakePHP-tutorials/MVC-Frameworks-Building-PHP-Web-Applications/315196-2.html (Accessed: 12 December 2016).

Summary of openEHR concepts:
https://code4health.org/platform/open_interfaces_apis/ehrscape/openehr_for_devs

Description of openEHR templates
http://www.openehr.org/releases/AM/latest/docs/OPT2/OPT2.html

A through description of API calls that are used to handle templates and compositions is provided in the following document:
https://www.ehrscape.com/reference.html#_composition

Examples of using EHRScape APIs:
https://www.ehrscape.com/examples.html

Description of openEHR concepts in the context of EHRScape:
https://s3-us-west-1.amazonaws.com/contattafiles/tnt3501/2e7cpBePqR2C21i/Think%21EhrWebTemplatesGuide.pdf

Guide to ReactJS development:
https://medium.com/@dabit3/beginner-s-guide-to-react-router-53094349669

Concept of presentation and container components:
https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0