Technical documentation

The Heureka! architecture of the Pool[2] requires to consider the concepts described in section architecture. For this purpose, we will describe needed technical documentation and best practices (as far as known) in this section.

One microservice, one application?

Even though it is quite common to implement one service as one monolithic application, this is not a required limitation. Since we are implementing different technical parts of a socio-technical collaboration system, we should consider up-to-date guidelines for RIAs (Rich-Internet-Applications). Thus, the core microservices developed for VcA itself consist of at least two applications: (1) a backend server system and (2) a frontend client application. While the first handles the connection to the database, OAuth2 integration, OES and implements some RESTful interfaces for data exchange, implements the client application a modern user interface considering requirements like Widgets. In the end all microservices consist of multiple (at least two) different software projects.

Additionally to the two base projects, there are several more possibly needed applications to implement a complete microservice. At first, all widgets are better isolated as distinguished software projects and repositories, since they have there own deployment flow and versioning using NPM. Furthermore, we have implemented some plugins to simplify the setup of backend applications. Those plugins are also separated from the core backend systems for the purpose of deployment and generalization of requirements.

The following subsections list all software projects for each microservice implement for Heureka! itself:

Drops

Category	Name	URL	Purpose
Backend	drops	https://github.com/SOTETO/drops	The server-side backend system written using Scala / Play 2.4.x.
Frontend	arise	https://github.com/SOTETO/arise	The client-side frontend system written using Vue.js.
Widget	vca-widget-user	https://github.com/SOTETO/vca-widget-user	A widget that implements user interface elements to search, select and show users. It is also written using Vue.js.
Plugin	play2-oauth-client	https://github.com/SOTETO/play2-oauth-client	A server-side backend plugin for Play 2.5 and Play 2.7 that serves as a OAuth2 client for the OAuth2 Provider that has been implemented in the backend of *Drops*.

Stream

Category	Name	URL	Purpose
Backend	stream-backend	https://github.com/SOTETO/stream-backend	The server-side backend system written using Scala / Play 2.7.x.

Category	Name	URL	Purpose
Frontend	stream-frontend	https://github.com/SOTETO/stream-frontend	The client-side frontend system written using Vue.js.

Microservice-architecture consists of microservices and nothing else?

Getting this motor running requires more than a set of loosely coupled microservices. According to the concept of Dynamic UI Fragment Composition every microservice needs to use some centralized UI elements, like a shared CSS library or a corporate navigation. Thus, we have introduced the following shared services and software projects:

Category	Name	URL	Purpose
CLI	heureka	https://github.com/SOTETO/heureka	Implements a CLI to deploy the Heureka! architecture and environment.
Dockerfile	grav-dockerfile	https://github.com/SOTETO/grav-dockerfile	A dockerfile used to deploy GRAV as part of the Heureka! environment.
Content	docu	https://github.com/SOTETO/docu	Contains the content of the Heureka! documentation and is cloned on every hosting system.
Backend	dispenser	https://github.com/SOTETO/dispenser	Handles a database to instantiate a navigation and hosts a shared CSS library that can be used by all microservices.
Widget	heureka-widget-navigation-2021	https://github.com/SOTETO/heureka-widget-navigation-2021	A widget that implements UI elements for a navigation. It is written using Vue.js.
Widget	vca-widget-base	https://github.com/SOTETO/vca-widget-base	A widget that implements basic UI elements that can be used by all frontend applications. It is written using Vue.js.
Backend	webapps-drops-oauth	https://github.com/Viva-con-Agua/webapps-drops-oauth	A server-side backend system that uses the play2-oauth-client to handle the OAuth2 based authentication of the Heureka! architectures shared session concept. It can be used for WebApps with no or external backend implementation to handle the authentication.

Communication between Microservices?

According to the concepts of Business Object Exchange, you can find here the technical documentation of all RESTful APIs and of the NATS service that is used to implement the OES.