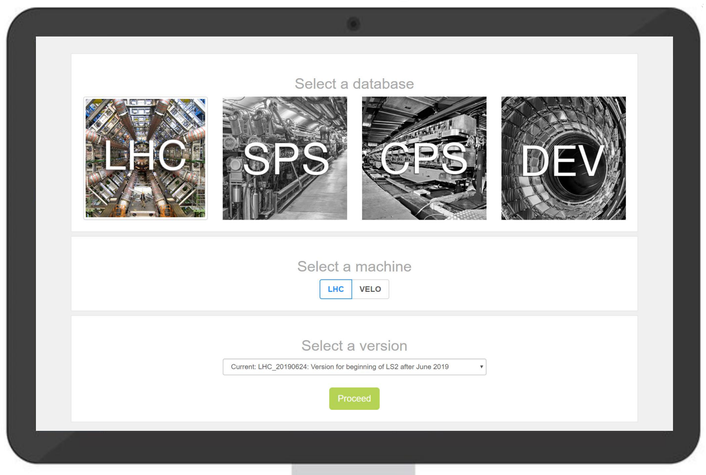
# VACUUM CONTROLS CONFIGURATOR: A WEB BASED CONFIGURATION TOOL FOR LARGE SCALE VACUUM CONTROL SYSTEMS

A. Rocha*, I. Amador, S. Blanchard, J. Fraga, P. Gomes, G. Pigny, P. Poulopoulou, C. V. Lima
CERN, 1211 Geneva 23, Switzerland

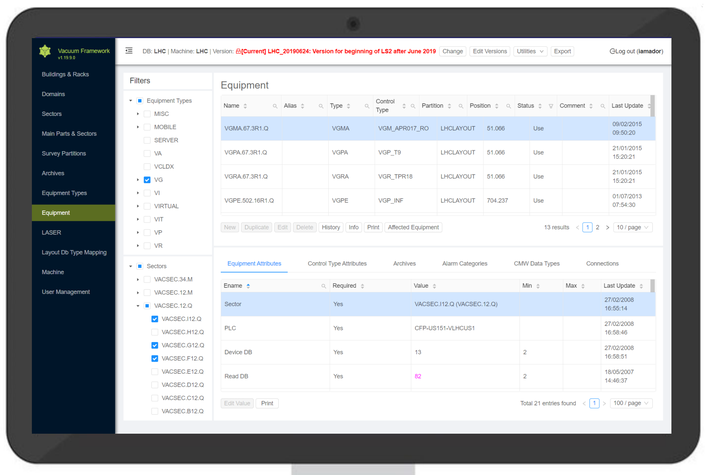
## ABSTRACT

The Vacuum Controls Configurator (vacCC) is an application developed at CERN for the management of large-scale vacuum control systems. The application was developed to facilitate the management of the configuration of the vacuum control system at CERN, the largest vacuum system in operation in the world, with over 15,000 vacuum devices spread over 128 km of vacuum chambers. It allows non-experts in software to easily integrate or modify the vacuum devices within the control system via a web browser. It automatically generates data for the configuration of the communication between vacuum devices and the supervision system, the generation of SCADA synoptics, long and short term archiving, and the publishing of vacuum data to external systems. VacCC is a web application built for the cloud, dockerized, and based on a microservice architecture. The application is divided into 4 microservices: *front end*, *validation & persistence*, *exporter*, and *synchronizer*.

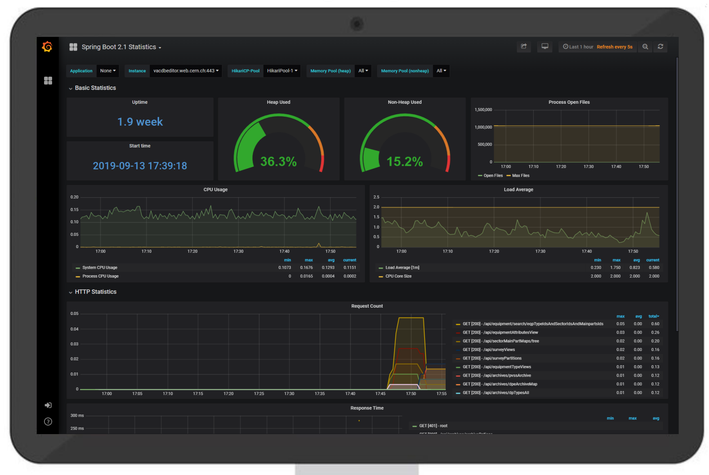
Home Page




vacCC User Interface




Application monitoring using Prometheus/Grafana




OPENSHIFT ROUTER

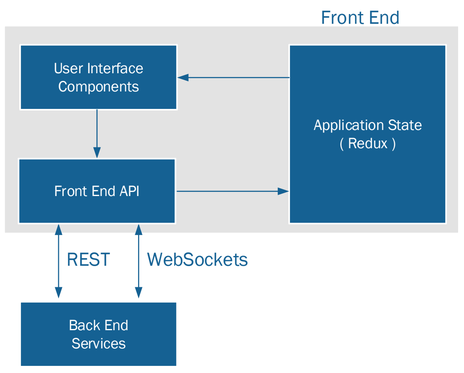
/ → LOAD BALANCER

/backend → LOAD BALANCER

/export → LOAD BALANCER

/synchronizer → LOAD BALANCER

## FRONT END

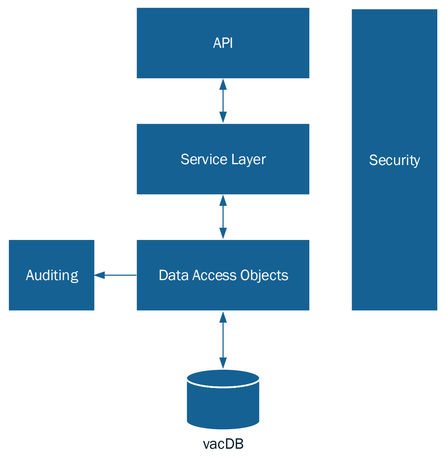



The front end microservice provides the user interface of the application. It allows users to be abstracted from the complexity of vacDB, enabling them to modify vacuum machine parameters that are required for the export of SCADA and PLC configuration files.

The single page application is organized following the React model, where web elements such as pages and their elements (buttons, tables, forms, etc.) are hierarchically organized into components. The components interact with backend services (validation & persistence, exporter, and synchronizer microservices) using REST and WebSockets. WebSockets are used in special cases of long lasting requests, like the ones in the *Exporter* microservice, that need to provide feedback to the UI.

Request data is stored in the application store, using Redux , that components can access directly.

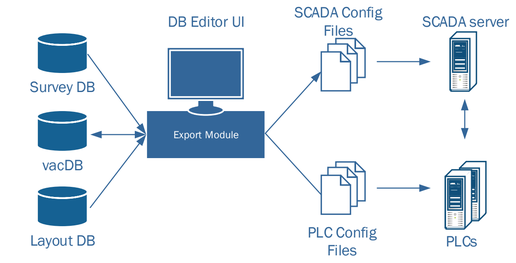
## VALIDATION & PERSISTENCE




The *validation & persistence* microservice is responsible for providing the interface between other microservices and vacDB. It achieves that by exposing RESTful APIs that allow other microservices to indirectly perform CRUD operations on the database.

- **API**: exposes REST endpoints for other services to interact with vacDB.
- **Service Layer**: handles requests from the API layer, performs data validation, and when necessary, combines results from multiple Data Access Object (DAO) operations to serve API requests.
- **Data Access Objects**: Provides objects that allow direct access to vacDB. This layer is implemented with Spring Data and Hibernate.
- **Auditing**: Records modifications in the configuration of the vacuum control system.
- **Security**: Provides authentication and authorization services for the entire application, ensuring that only users with the required privileges are able to perform database operations.

## EXPORTER

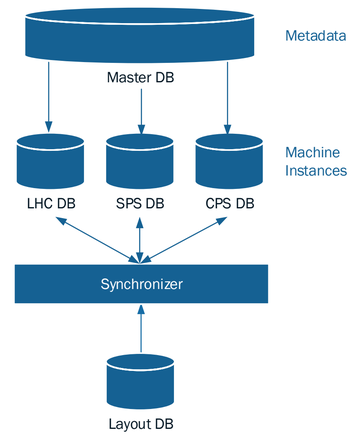



The *Exporter* microservice is responsible for generating the configuration files for both the PLCs and for the SCADA.

For each PLC, the exporter generates function block calls for each vacuum device connected to it, along with device datablocks; these contain all relevant information that will allow PLCs to connect and interact with device controllers.

For the SCADA, the exporter microservice generates configuration files with the data that will allow the configuration of all datapoints for every vacuum device. Each datapoint will be configured with the archiving settings defined in vacDB, and each datapoint element that requires communication with a PLC will be automatically configured to point to its corresponding memory location.

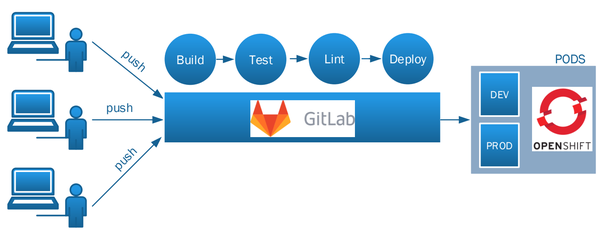
## SYNCHRONIZER




The purpose of the *synchronizer* microservice is to automatically import vacuum data from the Layout DB into vacDB, ensuring that the official, approved layout of the vacuum system is reflected in vacDB. The Layout DB is a CERN-wide database that models the architecture of CERN's accelerators. It contains data concerning most accelerator subsystems, including RF, beam instrumentation, magnets, cryogenics, and vacuum.

Users can trigger a differential analysis between vacDB and the Layout DB. The differences detected in the analysis are based on the create, update, and delete operations made on the Layout DB that are not reflected in the Machine DB, concerning vacuum sectorizations, and equipment and their attributes (position, type, and hierarchy). The analysis process provides users with a list of actions that need to be performed on vacDB to bring it up to date with the Layout DB. Users can use the synchronizer service to automatically perform the suggested updates.

## CONTINUOUS INTEGRATION AND DELIVERY




Every change made to the code repository passes through a pipeline that will build, test, and lint the code. In case of errors, the pipeline will stop and the developer will be alerted. Commits pushed to the master branch of the repository that pass the build, test, and lint stages are automatically deployed to the staging environment, a replica of production, where developers can perform additional testing. After validation in the staging environment, a tag of the master branch is created, and developers can trigger an automatic deployment to production.

## CONCLUSIONS

The front end and validation and persistence microservices of vacCC are in production since March 2019 and have completely replaced the vacDB-Editor as the tool for editing the configuration of the control system. Users have reported a significant increase in productivity using the new interface, which is especially important during the Long Shutdown 2 of the LHC, where tens of thousands of configuration changes are expected. We are currently in the validation phase of the exporter microservice, and we expect to complete the development stage of the synchronizer miroservice on late 2019.

The adoption of a microservices architecture in vacCC brought several advantages. It allowed to split a big problem into smaller, independent, and more easily manageable pieces of software, where software developers are able to work simultaneously in the different system components. Future upgrades of vacCC to new technologies can now be done on a service by service basis, without the need of a big team of software developers uniquely dedicated to upgrading the whole application at once.



*andre.rocha@cern.ch

Vacuum, Surfaces & Coatings Group
Technology Department