SUMMER STUDENT PROJECT REPORT
Developing and modernizing applications on the OP WebTools website

Student: Andrej Janchevski
University: Ss. Cyril and Methodius University, Skopje, Republic of Macedonia
Faculty: Faculty of Computer Science and Engineering
Supervisor: Dr. Emanuele Matli, BE-OP-ISO

Geneva, August 15, 2018
1 Introduction

The computer systems of CERN are not just composed of very specialized software that controls the complicated machines and hardware of the many physics experiments. Another great fraction of the software are web-based monitoring applications and services that provide simple user interfaces for directly viewing current experiment data, managing access permits to sites, planning shift schedules, logging events etc.

Specifically operators, i.e. employees of the OP group at CERN, as engineers are daily users of all of the services on the OP WebTools website, which as a hub provides easy access for users by linking to all of the available useful tools, each located on a separate website.

Sadly, a lot of the existing applications have become outdated with time - better development frameworks have emerged and are a viable replacement for the current design structure.

As such, it is up to the developers in the BE-OP department to update or in some cases completely redesign the front-end without having to change the underlying back-end and database logic.

In addition, a client from the operators may request a brand new application either providing currently requested functionalities or a way to simplify some existing procedure that requires use of more than one web tool.

The summer student project consisted of working on 3 different web applications: modernizing the existing call log and access requests web tools, and developing a new dashboard app that will provide useful information for operators in the TI island at the CCC.

The front-end rework consisted primarily on moving from the Angular to the Vue framework and using modern Node packages and JavaScript libraries for CSS styling, data formatting, response and request management etc. Existing well-documented back-end endpoints written in PHP were left mostly unchanged to preserve the logic and reusable previously-developed Vue components were utilized in order to standardize the appearance of OP WebTools across all apps.

Before going into detail into the development of each app, it is prudent to present the more important software tools used in this project and give a short introduction.

2 Software tools

2.1 Back-end and documentation

All of the REST API endpoints of the OP WebTools applications are implemented in PHP, with database queries usually written in plain SQL. The API is documented with Swagger annotations and can be tested and consulted through an automatically generated web interface.

2.1.1 PHP

PHP ("PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but much more can be done with PHP.[1]

In our case, PHP is used only to expose data with a REST API and not to generate HTML, which is completely dealt with in the front-end with JavaScript.

2.1.2 Swagger UI

Swagger UI allows a development team or end consumers to visualize and interact with the API’s resources without having any of the implementation logic in place. It’s automatically generated from the OpenAPI Specification, with the visual documentation making it easy for back end implementation and client side consumption.

OpenAPI Specification (formerly Swagger Specification) is an API description format for REST
APIs. An OpenAPI file allows to describe an entire API, including: available endpoints and operations on each endpoint, operation parameters, input and output for each operation, authentication methods, contact information, license, terms of use and other information. API specifications can be written in YAML or JSON. The format is easy to learn and readable to both humans and machines.[2]

In our case, a JSON file with the API specifications is generated from the Swagger annotations in the PHP code.

2.2 Front-end

Using the Vue framework with Semantic UI has become the new OP WebTools standard and the main goal is to update all of the tools to have a similar recognisable design and structure - reusing previously developed components and styles is encouraged. These and related software packages provide a state of the art look to the website and a modern and efficient way of building interfaces.

2.2.1 Vue.js

Vue is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.[3]

2.2.2 Semantic UI

Semantic is a UI framework designed for theming. Its key features are that it has more than 50 UI elements, more than 3000 CSS variables, 3 levels of variable inheritance (similar to SublimeText), it is built with relative length units for responsive design and it is flexbox friendly. Semantic allows developers to build beautiful websites fast, with concise HTML, intuitive JavaScript, and simplified debugging, helping make front-end development a delightful experience. Semantic is responsively designed allowing the website to scale on multiple devices. Semantic is production ready and partnered with many frameworks including Vue, which means it can be integrated with Vue to organize the UI layer alongside the application logic.[4]

2.2.3 Moment

Moment.js is a lightweight JavaScript date library for parsing, validating, manipulating, and formatting dates.[5]

2.2.4 Toastr

toastr is a Javascript library for stylized non-blocking notifications, such as success messages, warnings and errors.[6]

2.2.5 Lodash

Lodash makes JavaScript easier by taking the hassle out of working with arrays, numbers, objects, strings, etc. Lodash’s modular methods are great for: iterating arrays, objects, and strings, manipulating and testing values and creating composite functions.[7]
2.2.6 Highcharts

Highcharts is a SVG-based, multi-platform charting library. It makes it easy to add interactive, mobile-optimized charts to web and mobile projects. It features robust documentation, advanced responsiveness and industry-leading accessibility support.[8]

2.3 Deployment

In order to facilitate development the OP WebTools applications are deployed on three different servers: the development server is used frequently by the person/team currently working on an issue, the test server is used for demonstrating the application to clients and receive their feedback and the final stable version of the app is deployed on the production server.

A shell script utilizes tasks from a Gulp file to deploy a separate part of the source code(back-end, front-end or docs) to the respective server, i.e. 9 possible tasks in total, by providing arguments to the script. These 2 files can be reused in every web tool, by just making small changes to the Gulp file.

2.3.1 Gulp

gulp is an open-source JavaScript toolkit used as a streaming build system in front-end web development. It is a task runner built on Node.js and npm, used for automation of time-consuming and repetitive tasks involved in web development like minification, concatenation, cache busting, unit testing, linting, optimization, etc. gulp uses a code-over-configuration approach to define its tasks and relies on its small, single-purposed plugins to carry them out. The gulp ecosystem includes more than 300 such plugins.[9]

3 Projects

The summer student assignment consisted of developing, documenting and deploying 3 different applications. A form of Extreme Programming(XP), an agile software development methodology, was used in order to produce fast results in the short available time and frequent consultations with clients, supervisor and fellow students were common.

The call log and the tool for access requests were existing projects whose interface needed to be migrated from Angular to Vue while keeping the original functionalities. In addition, the call log was redesigned to work with a new database structure deployed by the Communications group after migrating to a new system. The TI dashboard was requested by a client as a new tool and as such had to be designed from the beginning, but with consideration to the specifications and requirements given by the client. All data was gathered by leveraging the existing REST APIs developed for previous apps.

In the next sections the specifics of each of the sub-projects will be presented while not going into the technical details of development and only showcasing the provided features of the different service.

3.1 Call Log

The call log application is used to monitor phone calls established from and to operators at the CERN Control Centre, in real time.

The design of the user interface is meant to be simple and concise in order to show only the useful information to the operator viewing it, while also being very efficient in tracking the current changes and updating as quick as every 5 seconds.

The new front-end is mainly composed of just 2 Vue components. Standardly, at the top of the page the reusable OP WebTools NavBar component is placed, modified with an added drop-down menu to select the relevant CCC island from one of the following: TI, LHC, SPS, PS or CRYO.
The **router** is used to redirect to the separate page for each of the sections, which can also be accessed from the browser. In addition, from the top bar users can choose whether they require the calls delivered during a certain date-time interval or the default option - retrieving the calls from the last 2 hours. The date interval can be selected by the provided **RangePicker** component, reused and slightly modified for this purpose.

Everything regarding the call logging is implemented in the **CallsTable** component which has a template based on table HTML stylized with Semantic UI. The table has 4 columns with the following information: calling number, called number, the deliver time of the call and its duration.

When a phone or mobile number is connected to some user account, the LDAP directory is queried by that number to retrieve the user details and format that information in the **PhoneToName** component - the user’s name, surname and department are displayed. The deliver time and duration are also formatted from the original ISO standard format to a human-readable date-time string. In addition, using other parameters, like the call’s start time, it can be extrapolated whether the call is still ongoing, has it been answered and was it an incoming or outgoing call.

As previously mentioned, efficiency is key for this application. Since updates have to be loaded real-time, a timer is used to send requests to the server every 5 seconds. Optimally, every call should be fetched only once, so in reality only the first request retrieves the whole batch of calls from the last 2 hours and the consecutive ones only check whether new calls have been delivered **after the deliver time of the latest call**. Unfortunately, in this manner an ongoing call’s final duration would never be retrieved so as an workaround it is allowed to fetch only the latest call if it is still unconcluded. If instead of fetching the recent calls a date-time interval is entered, the timer must be stopped and Vue should fetch the requested calls only once in this case as well.

### 3.2 Access Requests

Specialists of different teams might need to access CERN’s accelerators to check or maintain their equipment, but access to them is limited to a very few time slots during operation. Operators maintain a list of access requests in order to be able to optimise these limited time slots and organise access to the machines. The already existing access application provides this, but it’s interface had been in need of a rework using the new technologies.

Similar to other projects, a navigation bar is placed at the top of the page with this design containing a drop-down menu to select which section is in need of an access request and also to list the pending requests from that section. Similar to the call log, the section can be changed by routing to the separate page.

The request submission form aims to be as compact as possible. All of the necessary fields are present, providing many functionalities even with a reduced occupying space. The full form structure is contained within the **AccessRequestForm** component.

Every access request must contain:

- The full name and department of the requesting user, which can easily be obtained by searching the user database using a part of the person’s name - implemented using a modified **UserSearch** reusable component;

- The requested location, which can either be selected from a list of previous locations users have entered (sorted from most to least requested), or by manually entering a new location name;

- The duration of the request, with a choice of time unit and value, validated accordingly;

- Optionally, the user can write a description of the request and specify whether radiation protection is required during the maintenance;

Below the request form lies the **RequestsTable** showcasing the full information from a request, formatted for readability. An option to delete a request which has expired after the work has been
completed is also provided. As the goal of this tool is to be as simple to use as possible the main challenges lied in optimizing and modernizing the design, without any alteration of the existing server logic. On figure 1 a snapshot of the design is displayed.

3.3 TI Dashboard

The OP WebTools website provides a lot of useful services for the operators, aggregating data from different sources. It would be convenient if a new tool can gather and display much of this useful information for the TI operators, compressing the content and optionally linking to the original source for viewing the details. The Dashboard view’s main structure is a reactive grid, scaling and reorganizing with respect to the size of the page, composed of a navigation bar and 5 panels. The dashboard connects to many of the different web tools, each one of it’s panels updates in real time, can be viewed from any mobile of desktop device and was developed using a wide range of interface elements. On figures 2 and 3 the design of the dashboard can be viewed. The top menu links to portals, documents and online services useful to the TI operators. On smaller devices, like smartphones and tablets, the NavBar compresses to a single drop-down containing the links, to make better use of the limited screen real estate of mobile devices.

A list of recent events from the TI logbook is very practical data to display, which otherwise would have to be accessed separately. A tabular structure is utilized once more, with the EventsTable querying the logbook every 30 seconds for events either from the last Monday by default or from the user’s selected date. On mobile devices because of the screen size limitations a stylized list is used instead. Every event in this list can be double-clicked to display the full information in a pop-up screen.

A view of shift planning data for the current day would be a great convenience as well. In a separate panel, the current operator on duty is portrayed, with the full name and optionally a photo if provided, along with information about the designated supervisor and other operators on shift that day. This information can be modified in the planning tool and so the OperatorInfo component updates once a minute to account for eventual changes. Details about users, like department, building, email and phone numbers, can be obtained by searching by name in the top field - useful when a need to contact the operator on shift arises.

A lot of information can be summarized using statistics and the dashboard showcases 2 charts. Connecting to the call log, the first bar chart, constructed using the powerful Highcharts library, counts the number of calls delivered to and from the TI island every 5 minutes, during the whole day from midnight to midnight. The other graph is an existing frame built on top of the logbook which counts the number of alarms from the last 4 weeks using Elasticsearch. Finally, the operator can cycle through the collection of Vistar screens, each updating in real time with the frequency set to the screen’s refresh rate.

4 Conclusion

From the birth of the World Wide Web, CERN has set the standard of quality web development and is still a major source of innovation in this field. The massive CERN online network is run by a large quantity of software with some web services opened to the public and the others specialized for local use by employees. Therefore, it can be considered an honour to develop 3 tools that will be of assistance to the OP team and to have the unique opportunity to put into production something real, practical and immortalized while it is still displayed on a screen at the top physics institution in the world.
5 Appendix

New access request

<table>
<thead>
<tr>
<th>Date</th>
<th>Requester</th>
<th>Phone</th>
<th>Location</th>
<th>Time</th>
<th>Description</th>
<th>Radiation protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>27 Jul 2018</td>
<td>Andrej Janchevski</td>
<td>AD ACCESS</td>
<td>ATLAS CA998N</td>
<td>15 minutes</td>
<td>Just a quick look please. Please.</td>
<td>Yes</td>
</tr>
<tr>
<td>25 Jul 2018</td>
<td>Andrej Janchevski</td>
<td>ATLAS CA998N</td>
<td>LHCC TUNNELS</td>
<td>2 hours</td>
<td>Late night check.</td>
<td>Yes</td>
</tr>
<tr>
<td>25 Jul 2018</td>
<td>Andrej Janchevski</td>
<td>LHCC TUNNELS</td>
<td>ISOLDE CC</td>
<td>3 hours</td>
<td>Could be fun.</td>
<td>Yes</td>
</tr>
<tr>
<td>25 Jul 2018</td>
<td>Elemen S. Fasleski</td>
<td>ISOLDE CC</td>
<td>ISOLDE CC</td>
<td>4 hours</td>
<td>Test request.</td>
<td>Yes</td>
</tr>
<tr>
<td>25 Jul 2018</td>
<td>Elemen S. Fasleski</td>
<td>+41227670726</td>
<td>ISOLDE CC</td>
<td>4 hours</td>
<td>Test request.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Access requests from CPS

Figure 1: Snapshot of the access request interface

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Unit</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fri 27/01 12:15</td>
<td>Fault</td>
<td>gsd - CRAS-00009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed 27/02 17:25</td>
<td>Fault</td>
<td>colben - CDX106/SONEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed 27/02 17:30</td>
<td>Fault</td>
<td>colben - CDX106/SONEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri 24/02 10:46</td>
<td>Fault</td>
<td>traw - 271NNPVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed 25/02 10:30</td>
<td>Major fault</td>
<td>test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 16/09 16:14</td>
<td>Major fault</td>
<td>test Major TOC Tag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 16/09 16:27</td>
<td>Major fault</td>
<td>me again</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 16/09 16:11</td>
<td>Major fault</td>
<td>minor TOC tag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri 28/09 10:44</td>
<td>Fault</td>
<td>problem - CRAS-00015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri 29/09 11:10</td>
<td>Fault</td>
<td>test - CRAS-00015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon 03/10 09:40</td>
<td>Fault</td>
<td>test - BVORAHM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue 03/10 09:09</td>
<td>Fault</td>
<td>location - BVORAHM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 25/10 11:30</td>
<td>Fault</td>
<td>gds - ES1-ELANTRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 25/10 14:30</td>
<td>Ongoing work</td>
<td>Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu 25/10 14:41</td>
<td>Ongoing work</td>
<td>test 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed 06/11 09:02</td>
<td>Major fault</td>
<td>Major Power cut (check list)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed 06/11 11:07</td>
<td>Major fault</td>
<td>Major Power cut test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Snapshot of the dashboard - events table and shift planning
Figure 3: Snapshot of the dashboard - charts and Vistars

References


https://swagger.io/docs/specification/about/

https://vuejs.org/v2/guide/

https://github.com/Semantic-Org/Semantic-UI

https://github.com/moment/moment

https://github.com/CodeSeven/toastr

https://github.com/lodash/lodash

https://github.com/highcharts/highcharts

https://github.com/gulpjs/gulp