ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

DATA-CENTRIC WEB INFRASTRUCTURE FOR CERN RADIATION AND ENVIRONMENTAL PROTECTION MONITORING

Adrien Ledeul*, Catalina Cristina Chiriac, Gonzalo de la Cruz, Gustavo Segura, Jan Sznajd
CERN, Geneva, Switzerland

Abstract

Supervision, Control and Data Acquisition (SCADA) systems generate large amounts of data over time. Analyzing collected data is essential to discover useful information, prevent failures, and generate reports. Facilitating access to data is of utmost importance to exploit the information generated by SCADA systems.

CERN's occupational Health & Safety and Environmental protection (HSE) Unit operates a web infrastructure allowing users of the Radiation and Environment Monitoring Unified Supervision (REMUS) to visualize and extract near-real-time and historical data from desktop and mobile devices. This application, REMUS Web, collects and combines data from multiple sources and presents it to the users in a format suitable for analysis.

The web application and the SCADA system can operate independently thanks to a data-centric, loosely coupled architecture. They are connected through common data sources such as the open-source streaming platform Apache Kafka and Oracle Rdb. This paper describes the benefits of providing a feature-rich web application as a complement to control systems. Moreover, it details the underlying architecture of the solution and its capabilities.

INTRODUCTION

Radiation protection and environmental monitoring are fundamental aspects of the CERN Safety Policy. CERN's occupational Health & Safety and Environmental protection (HSE) Unit conducts a program in charge of monitoring the radiological and environmental impact of the organization. The aim is to ensure workplace safety for CERN employees and visitors, minimize the environmental impact of CERN, and provide regulatory authorities with comprehensive reports.

In order to achieve these objectives, a geographically distributed and heterogeneous set of instruments is continuously measuring the nature and quantity of ionizing radiations produced by the accelerators, possible contamination as well as conventional environmental parameters.

Radiation and Environment Monitoring Unified Supervision (REMUS) [1], based on WinCC Open Architecture (WinCC OA) [2] is the Supervision, Control And Data Acquisition (SCADA) system controlling this infrastructure. It is accessed from various control rooms across the Organization and has more than 200 active users.

At the time of writing, REMUS is interfacing 86 different types of devices. It contains 850 000 tags, manages 120 000 alarms and handles a throughput of 25 000 Input/Output operations per second. REMUS archives roughly 80 billion measurements per year.

One of the main challenges of such a system is to provide comprehensive yet accessible means to extract and exploit the data. REMUS itself allows users to display a large variety of synoptic views and control panels designed for the operation in control rooms. However, such user interfaces are not the most suitable for data extraction at a higher level of abstraction, and typically require physical access to terminals in a protected network.

Two use cases are particularly challenging to handle. The first one is that CERN radiation and environmental protection experts are in charge of transforming the data generated by REMUS system into business-specific reports. Such reports are used for further internal analysis and to consolidate CERN's communication with the host states and with the general public. The second one is to provide the CERN's Fire and Rescue service and other emergency response teams with access to near-real-time data on remote terminals. This is particularly useful at CERN, where installations for environmental monitoring are geographically scattered. The installation of dedicated monitoring screens is not always possible at the location where the access to the data is needed.

This paper describes the approach taken to make radiation and environmental monitoring data accessible to third party applications as well as why web technologies were chosen for the presentation of this data.

A DATA-CENTRIC APPROACH

This section introduces the approach taken for the consolidation and homogenization of the data layer.

The Data-Centric Manifesto

The data-centric mindset, as opposed to the application-centric one, considers the data to be the permanent assets, and the applications the temporary ones. The key principles, as expressed in the Data-Centric Manifesto [3], can be summarized as follows:

- Data is the key asset.
- Data is self-describing.
- Data is stored in non-proprietary formats.
- Data access control and security is the responsibility of the data layer itself.

This approach suits well REMUS use case, as radiation and environmental protection data must be kept for an indefinite amount of time, as requested by regulatory authorities. On the other hand, the applications publishing

* adrien.ledeul@cern.ch

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

and consuming the data change regularly, driven by technological evolution.

The key point is to think and design the data-set before designing the applications fulfilling the requirements. The concepts of a data-centric mindset applied to the CERN Radiological and Environmental Monitoring System are illustrated in Fig. 1.

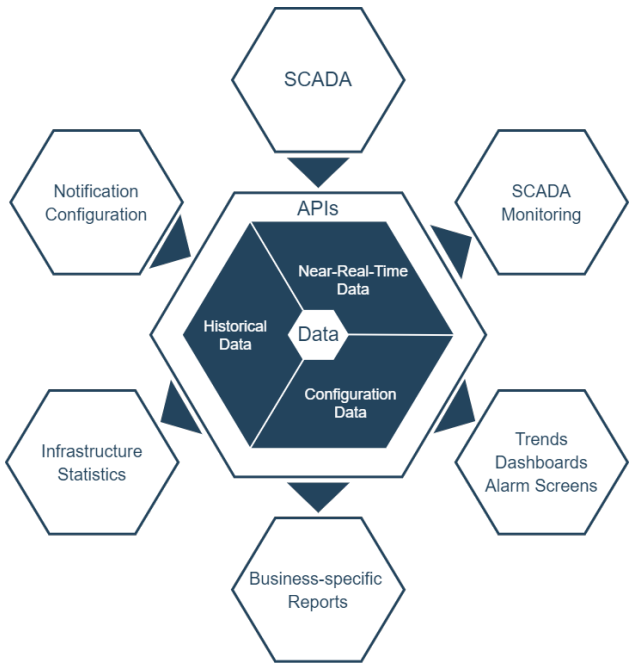

Figure 1: Data-Centric principles applied to REMUS. The center of the diagram represents all the data associated with radiation and environmental protection monitoring at CERN. The external blocks represents applications consuming and producing data through APIs.

The first steps taken in REMUS towards a data-centric approach were to consolidate and complete the data layer, as well as to unify the applications access to the multiple data sources of the control system through a common API.

REMUS DATA PIPELINE

This section describes the REMUS data flow, from the instruments to the data layer used by visualization tools.

Overview

REMUS is in charge of the data acquisition of over 5500 measurement channels, generating about 2600 data points per second in total. The acquired data is processed in parallel in two different ways, following the Lambda architecture [4]. On the one hand, data is processed in batch for long-term archiving. On the other hand, data is processed as a stream to allow immediate and continued exploitation [5]. The aim is to provide a data layer that can be homogeneously accessed through a data access API for historical and near-real-time data queries. The full data pipeline is illustrated in Fig. 2.

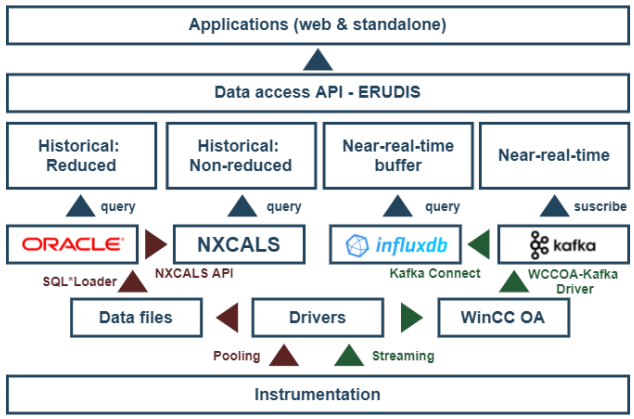

Figure 2: REMUS Data Pipeline. Applications consume data aggregated by the data access API. Data is fetched from various data sources with different latencies and time resolutions. Oracle and NXCALS are used for historical data; Kafka and InfluxDB for near-real-time data.

Batch Processing

REMUS fetches the data buffered in instruments' internal memories at a configurable polling interval. This data is written into files that are then injected in Oracle Rdb's partitioned tables using SQL*Loader [6] processes. A subset of the data is then transferred to the Next CERN Accelerator Logging Service (NXCALS) [7], based on Hadoop [8], where it is kept indefinitely at the highest resolution. The data stored in Oracle Rdb is reduced after two weeks, in order to guarantee low latency access.

Stream Processing

In parallel to the batch processing, REMUS receives continuous data streams from the instrumentation. The streams are transformed by the SCADA and published to a dedicated Kafka [9] topic using an open-source WinCC OA Kafka Driver [10], developed and maintained by CERN's HSE Unit. The topic is then filtered using Kafka Streams [11] to eliminate potential malformed messages. The processed stream is then sent to InfluxDB [12] via Kafka Connect [13] for temporary data retention.

Data Access API

Environment and Radiation Unified Data Integration Service (ERUDIS) is a data access API. ERUDIS provides high-level abstraction of the data sources that can be consumed by third party applications. The architecture of this API is based on Akka, an actor concurrency model framework, and the Alpakka [14] library, designed for handling data streams. These tools simplify the creation of interfaces for unifying heterogeneous data sources, as well as provide means for interactive management of data streams, such as buffering, advanced error handling and automatic redundancy mechanisms.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

REMUS WEB

This section introduces REMUS Web, a web infrastructure that allows users to visualize and extract near-real-time and historical data from desktop and mobile devices. REMUS Web takes advantage of the data pipeline introduced in the previous section to present the data to the user in a format suitable for analysis.

Rationale of the Approach

Web Applications Since its invention at CERN, the World-Wide Web [15] has evolved from a hosting platform for simple and static hypermedia documents to an infrastructure for the execution of complex applications [16]. In the last decades web applications became a popular and widespread solution in the software industry due to the advantages they present over standalone applications.

From the user's point of view, web applications are more accessible. They can be reached through the web browser and do not require complex installation processes or specific hardware configurations. Web applications work on any device capable of running a simple web browser, offering greater portability and cross-platform access. Furthermore, the requirements that end-user workstations must satisfy to run the applications are minimal, since computationally expensive tasks are delegated to the server.

From the developer's point of view, maintenance and update processes of web applications are simpler compared to standalone applications, since the application only needs to be deployed on the servers and no intervention is required on the end-user workstations. Web technologies are the most widely used and the most mature for the development of graphical user interfaces (GUI), which facilitates and speeds up development thanks to the large ecosystem, plugins and community around web development. In addition, web technologies are widely spread in the IT industry and it is easier to recruit professionals with web development skills than with SCADA and control systems skills. According to the 2021 Stack Overflow Developer Survey [17], web developer roles are the most popular on the market.

Progressive Web Applications In recent years the importance and impact of the web has been increasing. Most modern software companies base their business models on offering services through the web. This has meant the consolidation of the web as a service platform and the emergence of new web technologies that have made the line between web applications and native applications increasingly thin. In this model, cross-platform compatibility (desktop, mobile, and tablets) is essential to maximize the reach of the application and to provide the best user experience.

Progressive Web Applications (PWA) [18] provide users an experience on par with native applications. PWAs are a type of web applications that are intended to work on any platform using a standard-compliant browser, including both desktop and mobile devices. PWAs take advantage of recent advancements in web browser technologies such as service workers [19], web app manifests [20], and caching to bring features usually associated with native apps. These features include installability, responsiveness, network independence, re-engageability, and enhanced security. PWAs offer similar features to native and hybrid [21] applications, with a smaller bundle size and fast loading times [22]. Using PWA can significantly reduce the costs and resources required for the development and maintenance of a multi-platform application, since all platforms share the same codebase.

Data Accessibility The aforementioned features solve several of the challenges posed by accessibility of control systems' data. The use of a web application facilitates remote access to the data generated by the SCADA from any desktop or mobile device with a standard-compliant web browser. This is especially useful when operators must carry out interventions in remote facilities where there is no access to the control system terminals. Furthermore, REMUS Web is built as a PWA, allowing operators to access the data generated by every equipment from their mobile device. REMUS Web provides QR codes for every instrument connected to REMUS, which can be printed and attached to the instruments. In this way, when operators need to intervene on an instrument, they can access the instrument data in real time by scanning its QR code. This mechanism also allows operators in the field to access data from devices that do not have any integrated display.

Technological Stack

The technologies and tools used in the development of REMUS Web are listed below. These technologies have facilitated and accelerated the development of the tool, as well as improved its maintainability and scalability.

Front-end HTML, CSS and JavaScript are the standard technologies used for the development and design of web applications and therefore are the base technologies on which REMUS Web front-end is built. Additionally, high-level tools and libraries facilitate application development.

Most modern web applications are built on JavaScript frameworks for GUI development, which simplify the application development and maintenance processes. Currently, there are three major JavaScript frameworks that dominate the market: React [23], Angular [24], and Vue.js [25]. These frameworks have different characteristics and the decision of which one is the best option will depend largely on the circumstances of the project to be developed [26].

In the case of REMUS Web, we considered React to be the preferred option. React is a free and open-source JavaScript library for building GUIs which is developed and maintained by Facebook. React is flexible, efficient and declarative. According to the 2021 Stack Overflow Developer Survey [17], React is the most commonly used web framework and the most wanted by developers. At the performance level, React is fast and light. It offers a similar

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

performance to that of its direct competitors [27]. Since React is a UI library and not a complete framework, it offers a lot of flexibility and freedom to decide the architecture, tools and libraries used to build the application. Unlike other frameworks that use their own directives and templating languages, React's code is essentially pure JavaScript, which makes the learning curve shorter.

React has built-in functionalities for state management that are suitable for simple projects. However, the need for a more sophisticated state management solution arises as the complexity of the application increases and it becomes necessary to share state between different components. Redux [28] is the most widespread solution and the one used in REMUS Web. Redux is a predictable state container for JavaScript applications based on the Flux [29] architectural pattern.

Since one of the main goals of REMUS Web is to facilitate the study and analysis of the measured data, the use of a powerful and flexible charting library is of utmost importance. To this end, REMUS Web uses Highcharts [30]. Highcharts is a consolidated JavaScript SVG-based multi-platform charting library. It is flexible, well adapted to desktop and mobile devices and offers a wide variety of charts.

Real-time access to SCADA data away from the central control room is vital to assess and intervene quickly in critical situations. REMUS Web uses the WebSocket protocol [31] to establish a full-duplex real-time communication channel between the client and the server. In this way, the new data generated by the SCADA system is streamed to the client with a very low latency.

Back-end REMUS Web back-end is based on Java [32] and Spring Boot [33]. Spring Boot is an open source, microservice-based Java web framework. It is an extension of the well known Spring Framework [34], simplifying the set-up and configuration of standalone Spring applications and taking an opinionated view of the Spring platform and third party libraries to facilitate the development process.

Architecture

Figure 3 shows the architecture of REMUS Web. The application is deployed on the CERN Platform-as-a-Service (PaaS) infrastructure, based on RedHat OpenShift [35]. OpenShift is a container orchestration platform optimized for web applications. It allows building, testing and deploying web applications without provisioning and maintaining dedicated servers for each application. OpenShift runs the applications in Docker [36] containers.

The front-end of the application runs on the users' web browser and communicates with the back-end in two different ways depending on the nature of the information to be transmitted. WebSockets are used for the transmission of real-time data such as measurements or alarms. They are also used to send commands from the client to the server in order to start, stop or modify the parameters of the transmitted data flow. The rest of the communication between the client and the server is done through HTTP [37] requests.

The back-end of the application follows a classic three-layer architecture. The presentation layer exposes the REST and WebSocket APIs that the front-end consumes to interact with the back-end. The business layer contains the domain logic that drives the core functionalities of the application. Finally, the data layer is responsible for interacting with different data sources in order to persist and fetch data.

The data layer is based on Spring Data and ERUDIS. Spring Data is used to access and modify the data that is strictly owned by REMUS Web, such as reports, dashboards, or user preferences. In addition, it is also used to access the metadata of the SCADA system. On the other hand, ERUDIS is used to access both the historical and near-real-time data of the instrumentation connected to REMUS.

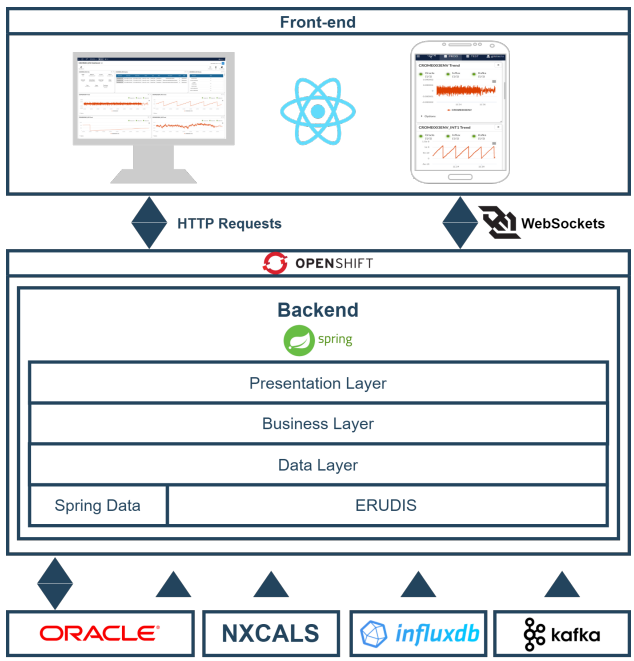

Figure 3: REMUS Web Architecture.

Functionalities

This section introduces some of the most important functionalities that are included in REMUS Web.

Notification Configuration Users can configure personalized e-mail and SMS notifications based on events produced by the SCADA. Users can select the notification sources, the event that triggers the notification (e.g. an alarm goes on or off) and the notification lifetime.

SCADA Monitoring The application allows the generation of custom reports to monitor the current and past status of the SCADA system. It is especially useful for analyzing incidents and to carry out investigations.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

Near-real-time Alarm Screen REMUS alarm screen is accessible directly from the web and displays the generated alarms in near-real-time. This eliminates the need to connect to the SCADA remote terminals in order to access the alarm screen, being accessible from any desktop or mobile device with internet access.

Near-real-time Trends REMUS Web allows to access and plot (see Fig. 4) both historical and near-real-time SCADA data. Users can fetch data from multiple data sources and define various parameters on the data to select. These parameters include time period, data resolution, and filters based on the alarm and fault status or the working mode of the instrument that generated the measurements.

Figure 4: REMUS Web Trends.

Domain-specific reports REMUS users require domain-specific reports for radiation and environmental protection. Such reports are used for internal analysis and to report to CERN's host states authorities. REMUS Web includes tools to generate these reports. Use cases include the extraction of aggregated radiation data and generation of the hyetographs (see Fig. 5).

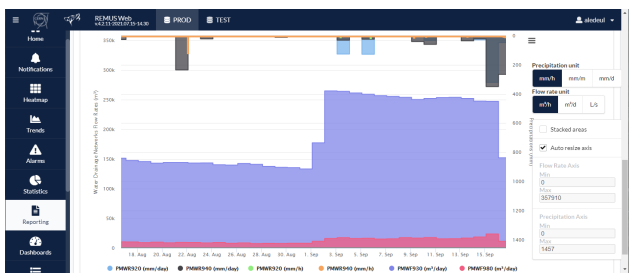

Figure 5: REMUS Web Reports. This screenshot shows a generated hyetograph, allowing experts to analyse the impact of rain on water flow rates.

Dashboards REMUS Web includes a powerful tool for creating and visualizing dashboards. The tool makes a wide variety of widgets available to users. Users can compose their own custom dashboards by combining different types of widgets. The layout and size of the widgets can be modified using a drag-and-drop interface. In addition, REMUS Web automatically generates a dashboard for each of the instruments connected to REMUS. Such dashboards show real-time data from this particular instrument, its alarms, and its parameters, as shown in Fig. 6.

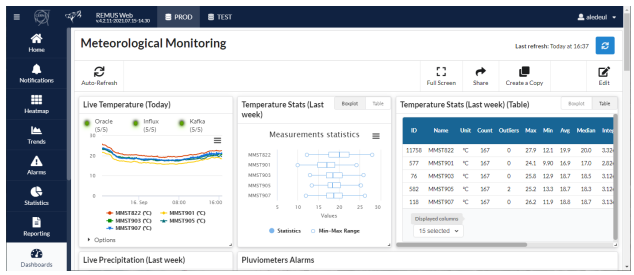

Figure 6: REMUS Web Dashboards.

Metadata Statistics REMUS Web allows access to the complete inventory of REMUS entities such as measurement channels, instruments, and events. Users can filter the data using multiple criteria and perform data aggregations to extract statistics and charts, as shown in Fig. 7.

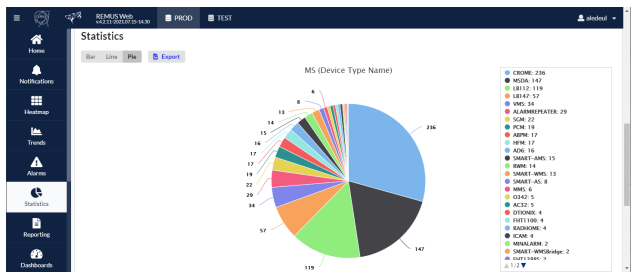

Figure 7: REMUS Web Statistics.

CONCLUSION

Control Systems' Human-Machine-Interface went a long way, from push buttons and wired lamps to modern SCADA graphical user interfaces. The fast evolution of data architectures and web technologies is an opportunity to go one step further in the accessibility of SCADA's presentation layer.

CERN's Radiation and Environmental protection SCADA extended its data extraction and analysis capabilities, exploiting a modern data-centric architecture and web technologies.

ACKNOWLEDGMENT

We would like to thank all the members, past and present, of the REMUS Project team as well as colleagues from Radiation Protection, Environment, Beams and Information Technology groups for their fundamental contribution to the success of the REMUS Web project.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV045

REFERENCES

[1] A. Ledeul, G. S. Millan, A. Savulescu, B. Styczen, and D. V. Ribeira, "CERN supervision, control and data acquisition system for radiation and environmental protection," in *Proceedings of the 12th International Workshop on Emerging Technologies and Scientific Facilities Controls (PCaPAC'18)*, 2019, p. 248.

[2] Wincc oa, https://www.winccoa.com

[3] Data-centric manifesto, http://datacentricmanifesto.org

[4] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 2785–2792.

[5] A. Ledeul, A. Savulescu, G. S. Millan, and B. Styczen, "Data streaming with apache kafka for cern supervision, control and data acquisition system for radiation and environmental protection," in *17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, 2019.

[6] J. Gennick and S. Mishra, *Oracle SQL* Loader: the definitive guide*. " O'Reilly Media, Inc.", 2001.

[7] J. Wozniak, C. Roderick, and S. R. WEPHA163, "Nxcals-architecture and challenges of the next cern accelerator logging service," in *17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA*, 2019.

[8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, Ieee, 2010, pp. 1–10.

[9] J. Kreps, N. Narkhede, J. Rao, *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011, pp. 1–7.

[10] Github repository for winccoakafkadrv, https://github.com/cern-hse-computing/WCCOAkafkaDrv

[11] Kafka streams, https://kafka.apache.org/documentation/streams

[12] Influx db, https://www.influxdata.com

[13] Kafka connect, https://www.confluent.io/product/connectors

[14] Alpakka, https://doc.akka.io/docs/alpakka

[15] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The world-wide web," *Communications of the ACM*, vol. 37, no. 8, pp. 76–82, 1994.

[16] S. Casteleyn, F. Daniel, P. Dolog, and M. Matera, *Engineering web applications*. Springer, 2009, vol. 30.

[17] Stack overflow developer survey 2021, https://insights.stackoverflow.com/survey/2021

[18] Progressive web apps, https://web.dev/progressive-web-apps

[19] M. Kruisselbrink, A. Russell, J. Song, and J. Archibald, "Service workers 1," W3C, Candidate Recommendation, Nov. 2019. https://www.w3.org/TR/2019/CR-service-workers-1-20191119/

[20] A. Gustafson, "Web app manifest - application information," W3C, W3C Note, Mar. 2021. https://www.w3.org/TR/2021/NOTE-manifest-app-info-20210324/

[21] A. Khandeparkar, R. Gupta, and B. Sindhya, "An introduction to hybrid platform mobile application development," *International Journal of Computer Applications*, vol. 118, no. 15, 2015.

[22] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive web apps: The possible web-native unifier for mobile development," in *International Conference on Web Information Systems and Technologies*, SciTePress, vol. 2, 2017, pp. 344–351.

[23] React, https://reactjs.org

[24] Angular, https://angular.io

[25] Vue.js, https://vuejs.org

[26] E. Wohlgethan, "Supportingweb development decisions by comparing three major javascript frameworks: Angular, react and vue. js," Ph.D. dissertation, Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[27] S. Krause, Js framework benchmark - round 8, https://stefankrause.net/js-frameworks-benchmark8/table.html

[28] Redux, https://redux.js.org

[29] Flux application architecture, https://facebook.github.io/flux

[30] Highcharts, https://www.highcharts.com

[31] I. Fette and A. Melnikov, The websocket protocol, 2011.

[32] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

[33] Spring boot, https://spring.io/projects/spring-boot

[34] Spring framework, https://spring.io/projects/spring-framework

[35] A. Lossent, A. R. Peon, and A. Wagner, "Paas for web applications with openshift origin," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 898, 2017, p. 082 037.

[36] D. Merkel *et al.*, "Docker: Lightweight linux containers for consistent development and deployment,"

[37] R. Fielding *et al.*, Hypertext transfer protocol–http/1.1, 1999.