CERN AppStore

Development of a multi-platform application management system for BYOD devices at CERN

Tamás Bató[1,*], *Sebastian* Bukowiec[1,**], and *Michal* Kwiatek[1,***]

[1]CERN

Abstract. The number of BYOD continuously grows at CERN. Additionally, it is desirable to move from a centrally managed model to a distributed model where users are responsible for their own devices. Following this strategy, the new tools have to be provided to distribute and - in case of licensed software - also track applications used by CERN users. The available open source and commercial solutions were analyzed and none of them proved to be a good fit for CERN use cases. Therefore, it was decided to develop a system that could integrate various open source solutions and provide desired functionality for multiple platforms, both mobile and desktop. This paper presents the architecture and design decisions made to achieve a platform-independent, modern, maintainable and extensible system for software distribution at CERN.

1 Introduction

A tool called Computer Management Framework (CMF) is used to manage computers at CERN. This tool was developed at CERN and is installed on every Windows device joined to the CERN domain. It allows two types of management approaches. Centrally managed computers where the schedule for application, updates, and patches are decided by the IT department and locally managed computers where a group of delegated administrators decides about which applications or patches should be installed on a defined set of computers and when it should happen. All of the managed devices must be a member of the CERN domain. The world is rapidly changing and also the expectations of the users at CERN. The Bring Your Own Device (BYOD) model has become very popular where people are being allowed to use one's personally owned device, rather than being required to use an officially provided device by the organization. Nowadays people own powerful and efficient devices, which, in most cases, are already part of their daily work routine. These devices are not part of the current management model, therefore they have no access to the CERN provided and licensed software.

To meet people's expectations CERN aims to extend the current offerings and providing a platform where recommended and required applications will be provided to all the personnel's devices regardless of them being a part of a CERN domain or not.

[*]e-mail: tamas.bato@cern.ch
[**]e-mail: sebastian.bukowiec@cern.ch
[***]e-mail: michal.kwiatek@cern.ch

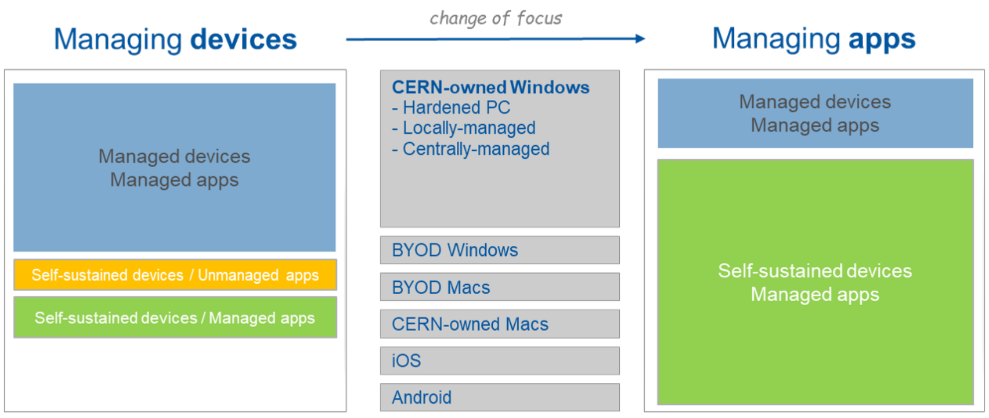


Figure 1. Shift in the device management model

2 Management model

Our new management model shifts the focus from managing devices to managing applications, as it is presented on Figure 1. In this way we are able to offer application support either to CERN owned or BYOD devices. To achieve this goal, we needed a new tool, which is able to provide applications without central management dependencies, like active directory membership, access to intranet or file shares.

As the list of supported (and planned to be supported) platforms became wider, we aimed for a system which is platform independent and can be made available on mobile and desktop environments as well. The designed tool strictly implements application management only. It does not manage the clients, push system updates or force any corporate policies.

Our goal was to build the system in a way to profit from already existing and widely used distribution tools. The provisioned applications come from different sources, which vary by platforms, as it can be a platform specific application store, an installer or a package for a platform specific package manager. The following section presents the selected technologies to implement the system.

3 System architecture

The system consists of multiple Docker [1] containerised components, which are automatically built by GitLab's continuous integration pipeline. Figure 2 presents the architecture of the system. The docker images of the front-ends and back-end are deployed to the Red Hat OpenShift container platform [2], while the MongoDB [3] database is running also as a Docker container but on a dedicated virtual machine in the CERN Cloud Infrastructure [4]. Every component is integrated with CERN's Authentication and Authorization system, which is a Keycloak [5] based solution.

3.1 Front-end

The user interface (UI) of the system is designed to be responsive. It supports smartphones, tablets and desktop clients. The React [6] library was used to implement the user interface, as it is an industry-standard and widely used open-source solution to build user interfaces. React

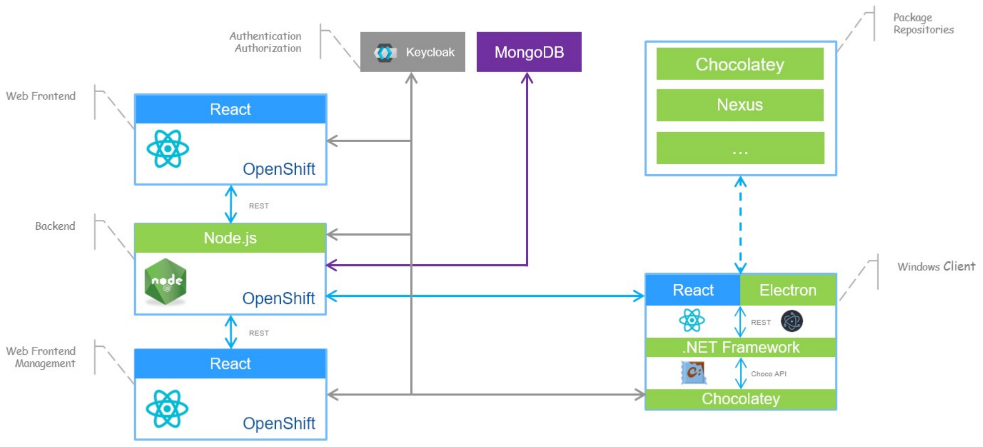


Figure 2. System architecture

is also widely used in the CERN IT department which allows providing required support and maintenance. Redux [7] is used as a global state management module. The user-facing components are built with Material-UI [8].

3.2 Back-end

We needed a back-end technology that is platform-independent, flexible and easily extensible. We decided to use Node.js [9] and Express to serve the REST API as it satisfied our requirements and made it possible to implement the back-end and front-ends with the usage of a homogeneous technology stack. The back-end is using a middleware based software architecture. The chained middlewares are responsible to filter out unauthorised requests (Keycloak middleware), check the received input, get requested data from the database, persist data into the database, and assemble the response to the clients.

The system used MongoDB as a data persistence layer during the development and pilot phase because it helped us in fast prototyping. MongoDB is a NoSQL database engine without strict data schema and it is able to store and return JavaScript objects as JSON documents, therefore the integration and usage in Node.js environment is straightforward. Regardless of MongoDB worked reliable during the pilot, the final system will use PostgreSQL [10]. PostgreSQL is officially supported by CERN's database team and it supports user-defined data types above the standard SQL data-types. In this way, PostgreSQL offers the same flexibility with these JSON typed fields as the NoSQL database engine did, but with all the advantages of the regular relational databases.

3.3 Desktop client architecture

The desktop client was designed to have the same look and feel on any platform. However, each platform has specific component requirements like package manager or libraries, therefore the client application consists of two separate layers, a platform-independent user interface, and a platform-specific package manager layer.

The user interface of the client is based on the same React UI as the web interface and it is packaged into an Electron [11] application. In this way, it has the same look-and-feel and code base as the web interface.

To manage applications the client needs a platform-specific layer. On the Windows operating system, this platform-specific layer is implemented in .NET C# and it is using Chocolatey [12] as a package manager. This layer has full access to the Windows API.

The communication between the two layers has to be implemented on a platform-independent way. We decided to use a RESTful communication interface between the Electron user interface and the package manager layer. The user can send commands from the React view through inter-process communication (IPC) to the Electron process to access OS level operations, or through RESTful requests directly to the package manager layer. It is a flexible solution where new platforms can be easily added to the system by implementing the platform-specific package manager layer of the new platform.

4 System functionality

4.1 Supported platforms

The system was designed to support mobile and desktop-like devices and to be easily extendable to the emerging platforms in the future. The current system supports Android and iOS mobile devices including Android and web applications for Chromebooks and has a dedicated desktop client for Windows operating system.

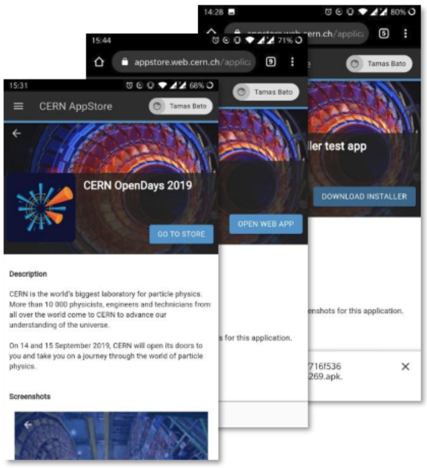
4.2 Application types

Besides native applications, the CERN AppStore provides also links to web applications, for example, CERNBox [13] and MapCERN. This allows us to provide in one place all types of applications that can be useful for our users.

The behavior of the system differs based on the application type. It can be:

- Redirection to a web URL
- Redirection to a platform-dependent application store
- Download an application installer (e.g. apk, exe)
- Install a package with platform-dependent package manager.

The users can benefit from the support of the communities and official application stores as well as from the offered and licensed software from CERN's own application repository.



CERN is the world's biggest laboratory for particle physics. More than 10 000 physicists, engineers and technicians from all over the world come to CERN to advance our understanding of the universe.

On 14 and 15 September 2019, CERN will open its doors to you and take you on a journey through the world of particle physics.

s for this application.

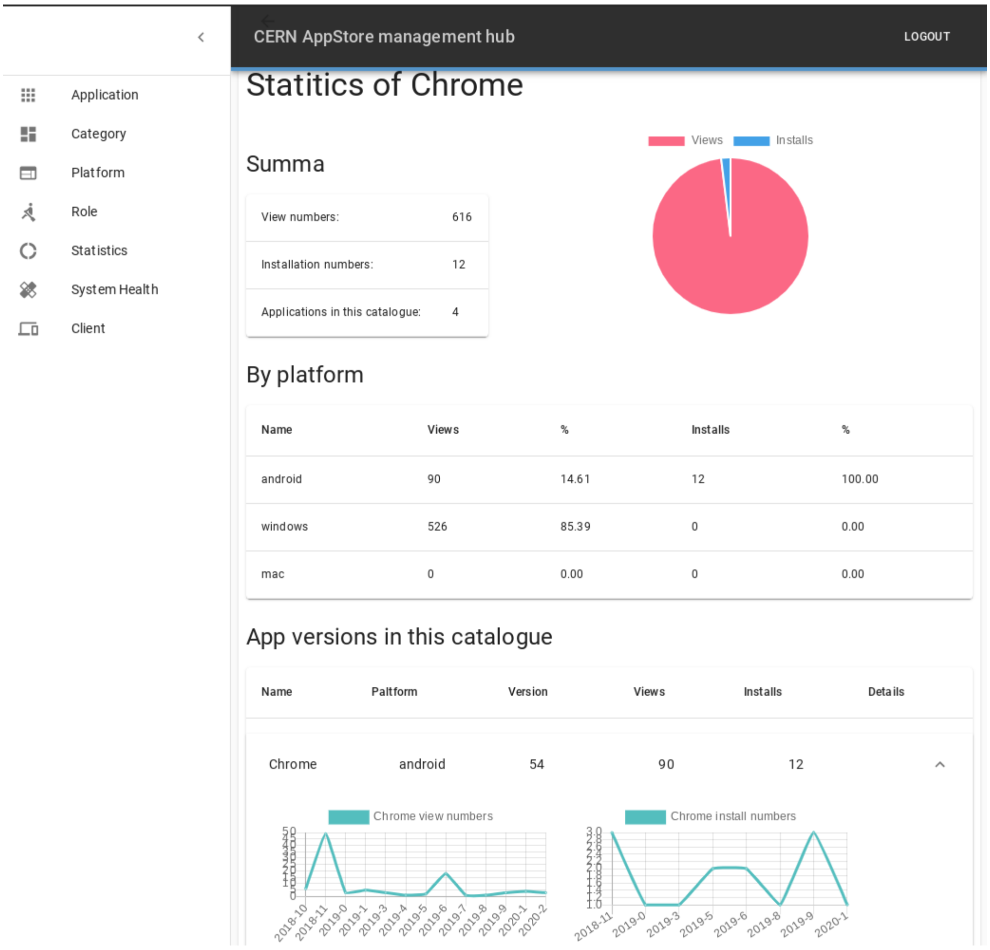
enshots for this application.

716f536
269.apk.

Figure 3. Store views on mobile devices

4.3 Categories and roles

The applications will be presented to the user by category. One application can be added to multiple categories. Access to a category can be restricted based on user roles. In this way, only a subset of the users will have access to a restricted category. Roles are assigned to users



Statitics of Chrome

Summa

View numbers:	616
Installation numbers:	12
Applications in this catalogue:	4

By platform

Name	Views	%	Installs	%
android	90	14.61	12	100.00
windows	526	85.39	0	0.00
mac	0	0.00	0	0.00

App versions in this catalogue

Name	Paltform	Version	Views	Installs	Details
Chrome	android	54	90	12	

Figure 4. Management interface, showing Chrome usage statistics

on the authentication system side (as roles in Keycloak) and will be included in the user's token after login. The system's back-end uses these roles to filter the application categories offered to a user.

4.4 Usage statistics

Collecting some basic usage data is necessary to analyse and understand the application usage and adjust the offered application's list to the user's needs. The system collects usage statistics from the clients and web users, like application page visits, installations, and application usage from native clients. Figure 4 presents an example view from the statistics calculated from the collected usage statistics.

5 Conclusion

The main challenge of the project was to design and build a system, which can operate on multiple platforms, both in mobile and desktop environments, offers applications from multiple application sources and eliminates all the organisation specific dependencies of the current central management framework. The system architecture is designed to be modular and containerised and the clients to be easily adaptable to new platforms.

The realised solution satisfies these requirements. It uses modern open source technologies and industry-standard solutions while it profits from already existing tools and the support of external and internal software providers.

6 Future plans

The system is currently under heavy development, in the future it will be extended with more supported platforms, for example, Linux. The data persistence layer will be changed from the currently used MongoDB. The move from NoSQL to PostgreSQL is already in progress.

After the base feature set of the system is implemented, the code is going to be released and the project will continue as an open-source project.

References

[1] Docker, *Container Technology*, https://www.docker.com/, Accessed: 2020-03-11

[2] OpenShift, *Containerization software*, https://www.openshift.com/, Accessed: 2020-03-11

[3] MongoDb, *Document-oriented database program*, https://www.mongodb.com/, Accessed: 2020-03-11

[4] Bell T, Bompastor B, Bukowiec S, Castro Leon J, Denis M, van Eldik J, Lobo M Fermin, Fernandez Alvarez L, Fernandez Rodriguez D, Marino A, Moreira B, Noel B, Oulevey T, Takase W, Wiebalck A and Zilli S 2015 Scaling the CERN OpenStack cloud, J. Phys.: Conf. Ser. 664 (2015) 022003

[5] Keycloak, *Single sign-on solution with Identity Management and Access Management*, https://www.keycloak.org/, Accessed: 2020-03-11

[6] React, *JavaScript user interface library*, https://reactjs.org/, Accessed: 2020-03-11

[7] Redux, *JavaScript application state library*, https://redux.js.org/, Accessed: 2020-03-11

[8] Material-UI, *JavaScript UI component library*, https://material-ui.com/, Accessed: 2020-03-11

[9] Node.js, *JavaScript runtime environment, Server-side JavaScript*, https://nodejs.org/en/, Accessed: 2020-03-11

[10] PostgreSQL, *Relational database management system* , https://www.postgresql.org/, Accessed: 2020-03-11

[11] Electron, *Desktop application framework, GUI with web technologies* , https://www.electronjs.org/, Accessed: 2020-03-11

[12] Chocolatey, *Package manager for Windows*, https://chocolatey.org/, Accessed: 2020-03-11

[13] Mascetti, Luca and Labrador, H Gonzalez and Lamanna, M and Mościcki, JT and Peters, AJ, Journal of Physics: Conference Series **664**, 062037, *CERNBox + EOS: end-user storage for science* (2015)