Magic Dashboard: Software System for Real-Time Development Tool Tracking

Daragus Botond*, Fodor Lóránt*, Kelemen-Fehér Dénes Bálint†, Mátis Szilárd-Gábor† and Sulyok Csaba*,
* Babeş-Bolyai University
Faculty of Mathematics and Computer Science
Romania, Cluj-Napoca 400084
† Codespring - Software development and outsourcing company
Romania, Cluj Napoca, 400664
Email: office@codespring.ro
* Email: daragusbotond@gmail.com, fodor.lori@hotmail.com, kelemen.balint@codespring.ro,
matis.szilard@codespring.ro and csaba.sulyok@gmail.com

***Abstract*—Nowadays, software developers are more and more involved in multiple projects simultaneously and their focus should be shared between each of them. The primary purpose of the Magic Dashboard project is to facilitate the working progress of these developers and project managers by keeping them up to date with the changes in their projects. The system displays data from different project management systems on screens that can be built into a wall or even behind a mirror.**

This paper presents the main functionalities and architectural structure of the Magic Dashboard project, detailing the technologies and tools used. In the second part of the paper, a comprehensive insight is given into how the application may be used in operation, and finally it mentions some of the future development possibilities.

I. Introduction

Nowadays, people have less and less time to spare as they try to keep pace with accelerated development. As a result, they try to find tools that can speed up and make everyday work easier. One of the best solutions to this problem is to create real-time feedback systems.

Such complex systems are developed and maintained by companies to track current states and conditions and display those on different types of dashboards. For example, a system called Valarm[1] can be used to prevent large fires, observe weather conditions, track down industrial machinery, etc. Another system that was created to facilitate everyday life is called MagicMirror[2]; it was one of the key pillars of the idea of the Magic Dashboard project. Because the need for productive timing in software development, the application has been invented and developed for this industry.

The primary goal of the Magic Dashboard project is to provide an application that can be used to save time for the workers in the field of software development. The developers of the project try to design an interface that is user-friendly, transparent and logical in both design and implementation.

The primary target audience of the Magic Dashboard software system are software developers and project managers. The software provides continuous monitoring and follow-up of activities on specific projects. This is accomplished by collecting the most important data that the user wants to display on a common interface. It is preferable to use a larger device for displaying the dashboard, whether it is a conventional or smart TV screen, projector or even a smart mirror built into a wall, depending on the amount of data the user wants to see. The shown data on the display is updated in real-time rather than being static and it is collected from various project management systems, such as GitLab or Jira. During the design phase of the software, it was important that the system be able to cooperate with the internal services of the company, therefore the system administrators have a number of configuration options. Also, the users can display different data on their dashboards according to their own privileges.

II. The Magic Dashboard project

A. Functionalities of the web application

The interaction between the users and the system is realized through a web interface, which can be accessed by both guests and authenticated users (see Figure 1). If the user decides not to log into the system, he/she can only access the view of the public dashboard, where the developer team's jointly assembled dashboard is displayed.

In case the user chooses to log in, it can be done by a GitLab validated login[3], whereupon the user can edit its own private, or even the public dashboard. By choosing the edit option, the private dashboard of the user will be displayed by default, initially providing a helper component that includes a small guide and the identifier of the dashboard as a QR code. In order for the users to be able to start fetching data from GitLab and to show them on the dashboard, they have to first set the access token provided by the GitLab service.

The dashboard is built up using many different components. Each displays one specific information for the developers. After placing components on the dashboard, the user can make settings for that specific one. For example, in the case of the 'Latest Contributors' component, the settings panel allows the following configurations: selecting the projects whose latest contributors' profiles the user wants to see; and also choosing the number of activities to be shown. This functionality can be

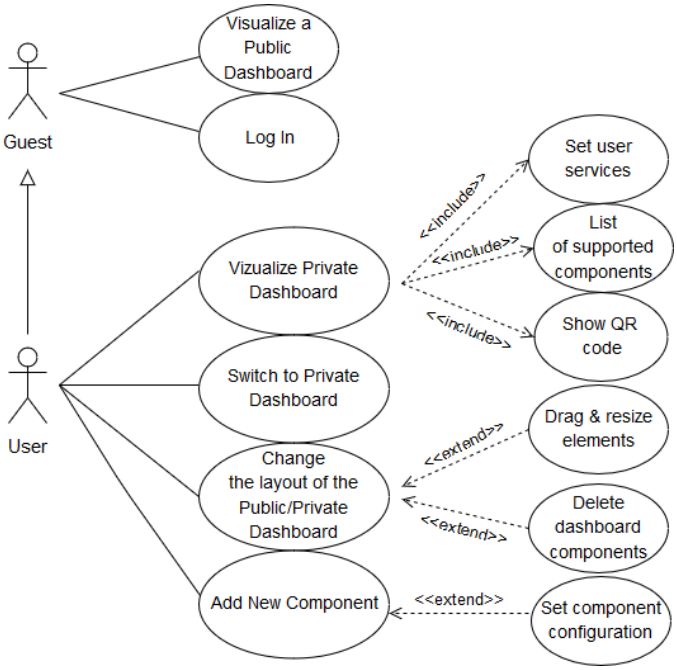

Fig. 1: Use case diagram of the web interface for clients and certified users.

performed on the editorial view of both the private and public dashboards.

The users also have the option to switch from the currently visible public dashboard to their own, private dashboard. This can be realized by either pressing the navigational icon on the web client, using the mobile application or by executing the appropriate Slack command [4]. At this point, the public dashboard is set for the user's private dashboard for a short period of time.

The public dashboard's view is designed for displaying on a mirror or a screen without any peripherals, therefore the user-defined components are still visible, but any interaction with them is disabled.

B. Functionalities of the mobile application

The Magic Dashboard system provides an option to switch between dashboards displayed on the web interface, which can be done by a mobile application running on Android or iOS devices. Since the results of the mobile application's functionalities are visible publicly on the web interface, the mobile and web clients are closely related. Using this application, a user can only make a private dashboard public if it is the legitimate property of the user.

The switching between dashboards can be performed by the users in two ways: they can either type the name of their dashboard into the application or read the QR code shown on the web interface using the integrated QR code reader. The users have the option to switch the public dashboard back as well, if the publicly shown dashboard is their private one. The names of the dashboards entered into the application get saved

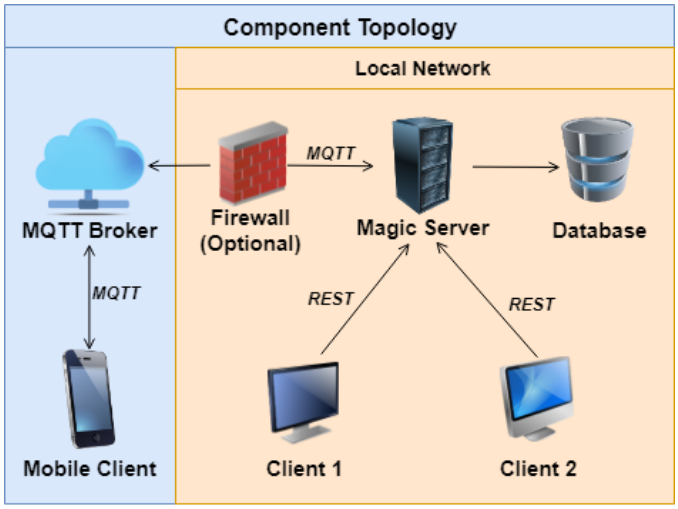

Fig. 2: Component topology of the Magic Dashboard project

into a list, so the users do not have to reenter those every time they want to perform a switch.

C. Architecture

The architecture of the Magic Dashboard system consists of four main components: a web server, a web client, a mobile client application and an MQTT [5] (Message Queuing Telemetry Transport) Broker. The topology of those components can be seen in Figure 2.

The framework of the architecture is greatly influenced by the environment that the system is deployed on, since the server and the web clients typically run on local (e.g. corporate) networks. These local networks are usually protected from requests from devices outside of the network by a firewall, so communication between the server and the users' smart phones could not be realized. This problem is solved by introducing a message-forwarding technology into the system: the MQTT Broker. The broker allows clients outside of the network to send their requests to this component first, and then forward it to a local network client, so the request is received by the server from a trusted device. Realization of such a communication does not require extra firewall rules, while neither the system nor the security of the network is compromised.

The components that build up the Magic Dashboard's system can be divided into additional subcomponents. Those and the connections between them can be seen in Figure 3.

The server is responsible for handling the data and serving the clients' requests, while also communicating with external systems. Structures in the Model package contain the representation of the central entities of the application. These structures are used directly by the data access (Store), Service and Controller layers. All the data of the application is stored in a document-oriented database.

The data access layer communicates with the database management system, inserting data into the database and retrieving from it. The service layer includes the implementation of the

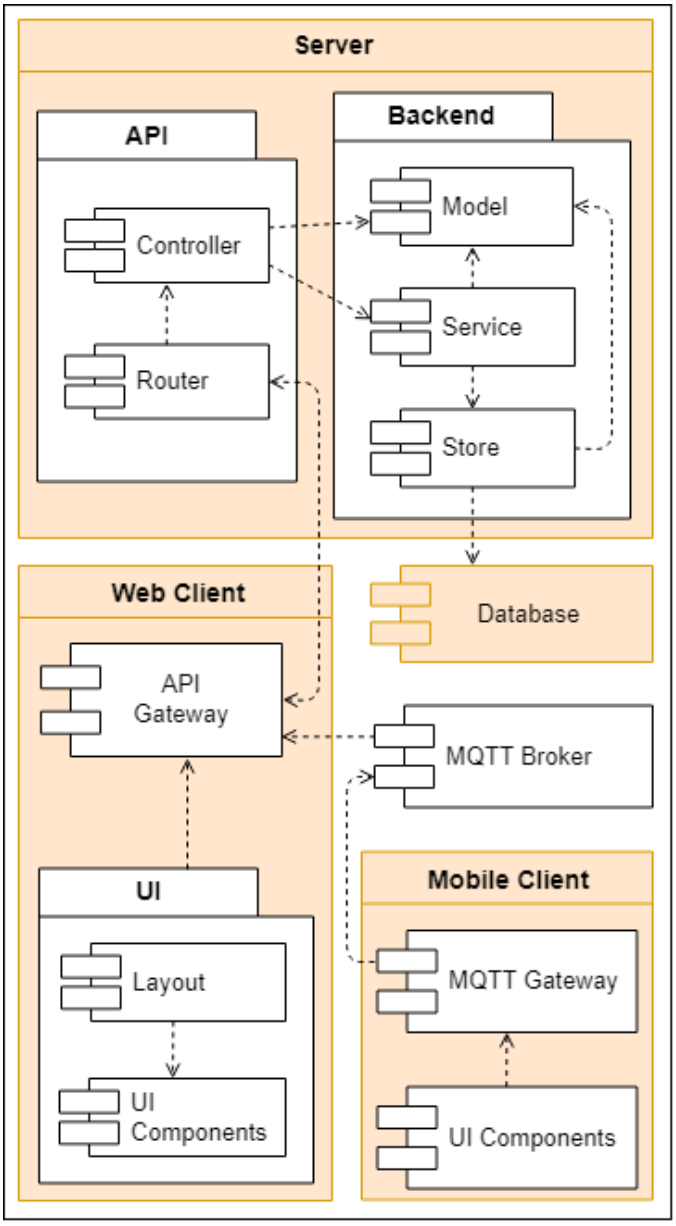

Fig. 3: Component diagram of the Magic Dashboard software system

operations that build up the logic of the application, which can be accessed through interfaces by the controller layer. The REST (Representational State Transfer) requests from the clients are received and forwarded to different controllers by the API Layer (Router). The controllers in cooperation with the service layers process the requests and return the required data to the users.

The web client application is responsible for displaying the graphical user interface. Similarly to the server, the structures of the entity representation for displaying the data can be found here. The communication with the server is provided by the API Gateway layer, whose methods are directly used by most of the components of the user interface.

The task of the mobile client includes displaying the user interface needed for publishing private dashboards remotely and to communicate with the MQTT Broker (and thus a web client on a local network). Methods for sending and receiving requests are implemented in the MQTT Gateway layer.

III. Technologies

The server side of the Magic Dashboard system is implemented in Golang and RehinkDB is used as database, the client side is based on the ReactJS and JavaScript technologies and the mobile application of the system was developed using React Native and Expo.

A. Golang

Golang[6], also known as Go, is an open source programming language developed by Google. When creating the language, the developers tried to eliminate problems and difficulties of other languages as much as possible, therefore Golang does not have exception handling, classes, dynamic types, but it provides, for example, lightweight thread management (goroutines) and memory management. Lightweight threads provide the ability to run competing and parallel tasks at the same time, and, contrary to C++, memory management also provides garbage collection. Its syntax is similar to the C language, it is compiled, so a performance-oriented code can be written with the help of Golang.

B. RethinkDB and GoRethink

RethinkDB [7] is a JSON document-based, open source, distributed NoSQL database that is optimized for scalable and real-time applications and provides real-time tracking of query changes. It uses the proprietary ReQL query language, which offers favorable chained queries. In the Golang programming language, the database operations with RethinkDB can be realized using the GoRethink [8] library.

C. Viper

For the Magic Dashboard to work, it is necessary to specify multiple configuration values, since the system is in contact with external services. This problem is solved by using the Viper[9] library developed in Golang. Various settings allow the application to load configuration information from environment variables, files (JSON, TOML, YAML, HCL and Java property files) or command-line arguments. Software developers can set default values or can define load priorities.

D. Gin

Gin is a HTTP web framework implemented in Golang, that, based on Go performance measurements[10], is currently one of the most effective and fastest-serving frameworks among its competitors. The significant difference in speed is due to HttpRouter, which uses the Radix tree [11] data structure.

E. ReactJS

ReactJS[12] or React is a JavaScript library that allows the developers to create single page application (SPA) interfaces. Its syntax can be either JSX[13] or simple JavaScript. The framework is based on components that can be any complex user interface elements. These components can be nested, communicating with each other as property-called data. Due to the nested features of the components, the user interfaces as components conform to the unit closure principle that a particular component is responsible for itself. Well-defined components can be used multiple times.

Another strength of React compared to other libraries is using the virtual DOM. This mechanism allows to refresh only those components which status has changed during use instead of refreshing the whole web page.

F. Golden Layout

Golden Layout[14] is a library written in JavaScript that can be used to modify the layout of a web application. The library allows the user to divide the surface of the web page into several smaller windows, or to add multiple tabs to the windows. These windows can be added from a side menu by drag-and-drop and are highly customizable.

For the Magic Dashboard, this multi-window splitting gives the users the ability to display different data of the components on different windows and to position them as they wish on the dashboard.

G. TypeScript

TypeScript[15] is an open source programming language that is a syntactical extension of JavaScript, developed by Microsoft. One of the most important features of TypeScript is that it provides classes, interfaces, primitive, generic and other types. JavaScript has no types and is an interpreted language, therefore many programming or syntax errors can not be filtered without running. TypeScript provides a static code analysis feature that displays errors during development.

H. React Native

The mobile client of the Magic Dashboard system is developed in React Native[16], a concept that is based on ReactJS. Upon compilation of a React Native application, native code is generated for different mobile platforms and in some cases, developers need to make platform-specific settings or write platform-specific code. A major advantage of React Native to ReactJS is that new component types can be used exclusively for mobile applications. The Magic Dashboard project seeks to reach mobile applications on as many platforms as possible.

I. Expo

Expo[17] is a toolkit built around React Native so that software developers can simultaneously build Android and iOS applications. It has the advantage of integrating the native code with the root directories so that software developers do not need to develop separate Java/Objective-C native modules. Expo provides an opportunity for developers to make the application under development available regardless of the environment, thus facilitating the tracking of their status and performing testing on multiple devices.

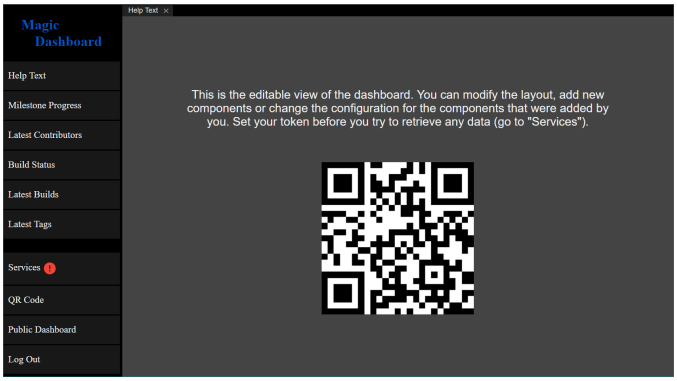

Fig. 4: The editorial view of the dashboard after signing in to the system

J. Git and GitLab CI

During the development of the project the Git[18] was used as a distributed version control system. The new functionalities were always implemented on a new branch, following the rules of 'Git-flow'[19].

GitLab CI[20] has been used for continuous integration, where every new modification on the code (push) was built for both server and client applications. GitLab CI allows for automatic containerization as well. The Magic Dashboard project creates a Docker image on both client and server pages for each tagged state, and it will be uploaded to a central Docker registry. As a last step, the system is deactivated automatically to a test server.

IV. Using the Magic Dashboard

A. The web client

By entering to the web application for the first time, the user is presented with two options: he/she can display the public dashboard or sign in to the application by clicking the "Login" button. If the latter option is chosen, the browser redirects to the GitLab web page where the user must agree that the application may use the Gitlab API with his/her rights. If this is done, the user will be successfully logged in to the Magic Dashboard system and the browser will return to the above-mentioned interface. Here, instead of the "Login" button, the "Edit Dashboard" button will be displayed. Choosing this option allows the user to enter the editorial interface of the dashboards, where a single component called 'Help Text' can be found, that contains a short guide and the QR code of the dashboard (see Figure 4).

On the left side of the interface the menu panel is located. The top side of the menu contains all the possible components that can be placed to the dashboard, while the bottom part of the menu includes the 'Services', 'QR Code', 'Preview' and 'Logout' buttons, respectively. By clicking the 'Services' button, the user may set the access token of his/her GitLab

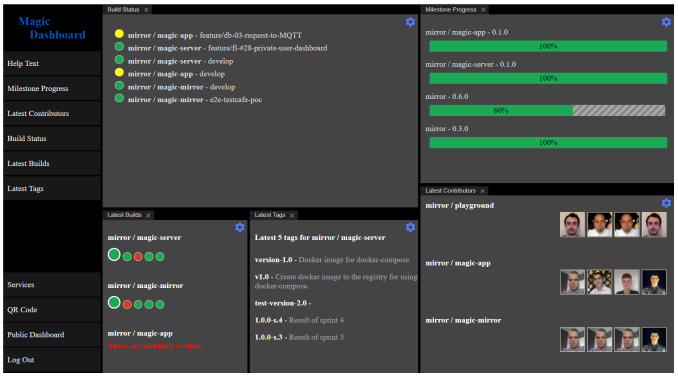

Fig. 5: Example view of a fully configured dashboard

account. Without doing so, the system will not be able to request data, so if this token is not set, a red exclamation mark will show up on the mentioned menu button. By selecting the 'QR Code' option, the current dashboard's QR code will be displayed. The 'Preview' button will show the current dashboard's view where no interaction is possible with the dashboard, or by clicking the 'Log Out' button the user can log out of the application. On the right bottom side of the interface, a switcher icon is visible that can be used to change between the editing view of the user's private and the public dashboard.

Placing the components in the menu on the dashboard is done by the drag-and-drop method. The interface gives opportunity to scale and move components on the dashboard freely, and the users also have the option to close the components at any time. By clicking on the gear icon in the top right corner of a component, a pop-up window displays the settings for that component. After configuring the components, the dashboard looks similar to the one shown in Figure 5.

B. The mobile client

Using the Magic Dashboard system's mobile application the user can make his private dashboard public remotely, without interacting with the web interface. To do this, the application has to know the private dashboard's identifier. This can be set on the welcome screen of the application: the user can choose to scan the QR Code shown on the web interface of his/her private dashboard by clicking the 'QR Code' button or to enter the name of it by pressing the 'Name' button. After the user gives the desired ID, it is possible to publish the dashboard for a few minutes by pressing the red button in the middle of the display. The user has the possibility to switch back to the public dashboard without waiting for the automatic change to happen by pressing the same publishing button again. The aforementioned views of the mobile application can be seen in Figure 6.

The application allows the user to store multiple dashboard IDs. These identifiers can be scanned in the publishing view, where the QR code can be scanned or the name of the dashboard can be typed. The desired dashboard to publish can be chosen from a drop-down list.

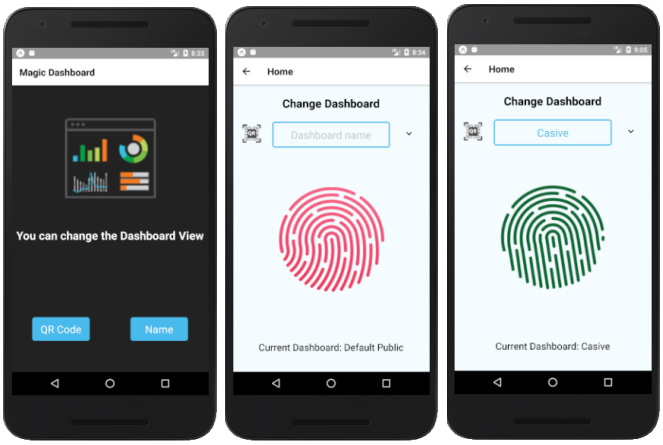

Fig. 6: The user interface of the mobile application. The first picture shows the Home Page, where the user can choose how he/she wants to change the dashboard view, by scanning a QR Code or by typing the name of the dashboard. In the second picture an input field and a button is shown, where the expected text is a name of a private dashboard and by tapping the red icon the user can change the dashboard view. In the last image the green button indicates that the dashboard has changed successfully.

CONCLUSIONS AND FUTURE WORK

Within the framework of this project, the Magic Dashboard managed to become an application that can be used by software developers and project managers for tracking the current state of their projects in real-time, on a single dashboard. Beside the web client, a mobile application has also been developed, which helps the users to remotely make their private dashboard public.

During the development, many ideas have arisen that could expand the functionalities of the application, such as integrating other project management systems and development tools (Jira, GitHub, etc.) or supporting more than one private dashboard per user. Creating new components to display different data from supported project management systems and upgrading existing ones (e.g., a component that shows the latest contributions for a GitLab project should also show the type of the contribution) is also a potential development option. Currently the web client uses a long polling mechanism to retrieve data from the server, but it would be an improvement if it used an event-oriented solution instead.

For controlling the behavior of the dashboard, IoT (Internet of Things) devices could be integrated into the application, such as LED strip lights (that flashes, if something changes on a component) or physical 'doButton' (that will make the user's private dashboard as public for a short amount of time by pushing it). Such controlling can be realized by implementing it through virtual personal assistants (e.g., Amazon Alexa, Google Home, Microsoft Cortana, Samsung Bixby).

REFERENCES

[1] "Valarm monitor anything, anywhere." [Online]. Available: http://www.valarm.net

[2] "The open source modular smart mirror platform." [Online]. Available: https://magicmirror.builders/

[3] "Gitlab as an oauth2 provider." [Online]. Available: https://docs.gitlab.com/ce/api/oauth2.html

[4] "Slash commands." [Online]. Available: https://api.slack.com/slash-commands

[5] A. Banks and R. Gupta, "Mqtt version 3.1. 1," *OASIS standard*, vol. 29, 2014.

[6] A. A. Donovan and B. W. Kernighan, *The Go programming language*. Addison-Wesley Professional, 2015.

[7] G. Tiepolo, *Getting started with rethinkdb*. Packt Publishing Ltd, 2016.

[8] "Gorethink." [Online]. Available: https://godoc.org/github.com/GoRethink/gorethink

[9] "Managing configuration with viper." [Online]. Available: https://scene-si.org/2017/04/20/managing-configuration-with-viper/

[10] "Gin - package httprouter." [Online]. Available: https://github.com/gin-gonic/gin/blob/master/BENCHMARKS.md

[11] R. L. Angle, E. S. Harriman Jr, and G. B. Ladwig, "Radix tree search logic," Feb. 16 1999, uS Patent 5,873,078.

[12] A. Fedosejev, *React. js Essentials*. Packt Publishing Ltd, 2015.

[13] "Jsx documentation." [Online]. Available: https://jsx.github.io/doc.html/

[14] "Golden layout documentation." [Online]. Available: http://golden-layout.com/docs/

[15] G. Bierman, M. Abadi, and M. Torgersen, "Understanding typescript," in *European Conference on Object-Oriented Programming*. Springer, 2014, pp. 257–281.

[16] B. Eisenman, *Learning React Native: Building Native Mobile Apps with JavaScript*. " O'Reilly Media, Inc.", 2015.

[17] "Introduction to expo." [Online]. Available: https://docs.expo.io/versions/latest/

[18] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.", 2012.

[19] D. Kummer, "Git-flow cheatsheet." [Online]. Available: https://danielkummer.github.io/git-flow-cheatsheet

[20] "Gitlab continuous integration and deployment." [Online]. Available: https://about.gitlab.com/features/gitlab-ci-cd/