



The process of building an admin dashboard user interface

Duong Thien Ly

Bachelor's Thesis
Degree Programme in BIT
2020



Author(s) Duong Thien Ly	
Degree programme Business Information Technology	
Report/thesis title The process of building an admin dashboard user interface	Number of pages 41
<p>MariaDB Corporation Ab is a global software vendor that develops and contributes to the well-known open source MySQL database which was forked and managed by the MariaDB Foundation. The company is specializing in developing solutions arounds MariaDB server. MaxScale, a database proxy open source project which is one of the core components of MariaDB Platform helping to extend the availability, scalability and security of MariaDB Server. MaxScale provides a command line administrative client tool called maxctrl that internally uses REST API to configure MaxScale at runtime.</p> <p>This thesis is established due to the need of developing an alternative solution to maxctrl which is a web browser application that operates as an admin dashboard user interface. The application should allow to configure MaxScale in a visually appealing, intuitive and user-friendly way. It will be built by using Vue.js framework along with its support plugins and libraries such as Vue Router, Vuex, Vuetify and so on.</p> <p>Though this is a graphical user interface product-oriented thesis, it will not include all development processes in terms of user interface such as prototype designs, user experience designs and user story due to confidential information of the company. Therefore, the primary objectives of this thesis can be divided into three categories comprise of setting up developer's working environment, improving MaxScale REST API for using in web application in terms of MaxScale user authentication, implementing the authentication user interface page.</p>	
Keywords Javascript, Vue.js, Vuex, Vue Router, Docker, MaxScale, MariaDB Server, REST API	

Table of contents

Terms and Abbreviations	1
1 Introduction	2
1.1 Thesis structure	2
1.2 About the company	3
1.3 Objectives and scope	4
1.4 Support from the commissioning party and copyrights	4
2 Theoretical framework.....	5
2.1 Vue.js	5
2.1.1 Reactivity system in Vue.js.....	6
2.1.2 Vue instance lifecycle hooks	9
2.2 MariaDB Platform.....	12
2.2.1 MariaDB Server relational database	12
2.2.2 MariaDB MaxScale.....	13
2.3 Docker.....	15
3 Empirical.....	17
3.1 Initializing the project	17
3.1.1 Setting up MaxScale environment	17
3.1.2 Vue CLI and Visual Studio Code configuration.....	22
3.1.3 Setting up necessary Vue.js plugins	26
3.2 MaxScale REST API authentication approach	28
3.2.1 Current authentication method.....	28
3.2.2 JWT for SPA.....	29
3.2.3 Storing the token	30
3.3 Implement the graphical user interface of authentication page.....	33
3.3.1 SSL encryption	33
3.3.2 Login page	35
4 Discussion	40
4.1 Problem encountered	40
4.2 Limitation of researched authentication approach	41
4.3 Further research	41
4.3.1 Summary	41
References.....	42
Table of figures	48

Terms and Abbreviations

SQL	Structured Query Language
DMBS	Database Management System
DOM	Document Object Model
API	Application Programming Interface
SASS	Syntactically Awesome Style Sheets
CSS	Cascading Style Sheets
GUI	Graphical User Interface
CLI	Command Line Interface
SSL	Secure Socket Layer
TLS	Transport Layer Security
HTTP	Hypertext Transfer Protocol
Database Proxy	A middle layer that locates between the database server and the application
REST	Representational State Transfer
VM	Virtual Machine
SPA	Single Page Application
MITM	Man-in-the-middle attack
XSS	Cross-site-scripting
CSRF	Cross-Site Request Forgery
CA	Certification Authority
JWT	JSON Web token
GIT	A version control system

1 Introduction

MaxScale, a core component of MariaDB platform which acts as a database proxy helping to extend the availability, scalability and security of MariaDB Server. At the moment, MaxScale provides a command line interface administrative tool called maxctrl which internally uses REST API to configure MaxScale at runtime. (MariaDB Corporation Ab 2020). The product goal behind this thesis project is to provide web-based application with similar functionalities to maxctrl CLI tool. The advantage of using a web application is to configure MaxScale at runtime in a visually appealing, intuitive and user-friendly way rather than using a CLI which requires tremendous time of memorizing the commands and syntax. Because of confidential information relating to MariaDB product design system, only part of the application which is the authentication page in terms of user interface will be included in this thesis.

1.1 Thesis structure

Overall, the structure of the thesis can be divided into four chapters:

Firstly, the introduction which provides the summary information about the company, objectives of this thesis and commission party information in terms of support and copyrights.

Secondly, the theoretical part of this thesis provides necessary knowledge and explanation behind the empirical part consisting of the UI framework called Vue.js, MariaDB platform and Docker (a tool to containerize application).

Thirdly, the first chapter in the empirical part describes the process of initializing the project which includes the steps of building, installing MaxScale; the setting of multiple MariaDB relational database servers. In addition, the setting up of Vue.js development environment comprising of Visual Studio Code as a coding editor and several libraries as Vue.js' plugins. The second part focus mainly on the improvement approach to the current REST API authentication in MaxScale along with the implementation of the graphical interface for user authentication page using Vue.js.

Finally, the conclusion part evaluates the successfully of the thesis implementation as well as difficulties encountered while developing the application.

1.2 About the company

MariaDB Corporation Ab was originally named as SkySQL Corporation Ab which was founded in 2010 by one of the founders of MySQL Ab, Michael “Monty” Widenius. He is known as the main author of the well-known open source MySQL relational database and one of the founding members of MySQL Ab (Widenius, M. 2014).

MySQL Ab was then acquired by Sun Microsystems in 2008 as for the believe in the open source approach that Sun Microsystems had made with their products before acquiring MySQL Ab (Widenius, M. 2008). Notwithstanding, in 2010, Oracle finally acquired Sun Microsystems includes the open source database MySQL. As for the concerns of keeping the commitments to open source as well as the leaving of MySQL developers at Sun Microsystems after the acquisition which may results in the hold back of the development and support for MySQL, Michael “Monty” Widenius employed the MySQL core development team at his Monty Program Ab company to work on MariaDB which is a forked of MySQL.

After the acquisition, in 2010, SkySQL Corporation Ab was founded in order to employed support team, consultants, trainers and salespeople from MySQL Ab to provide support and other services around MySQL and MariaDB (Widenius, M. 2010). SkySQL operates as a partner for Monty Program Ab since then before it eventually acquired Monty Program Ab and joined the MariaDB Foundation in 2013. In 2014, SkySQL changed its name to MariaDB Corporation Ab as the company “*is the main driving force behind the development of the MariaDB server and the biggest support provider for it* (Widenius, M. 2014).

At the moment, MariaDB Corporation Ab contributes the most to the MariaDB Foundation and has largest number of MariaDB experts. The company provides subscription services and additional enterprise features around MariaDB which includes MariaDB platform, MariaDB Cloud, MariaDB Platform Managed Service and ClustrixDB. The company has two headquarters which are located in United States and Finland as well as other representative offices in Europe, Asia and Americas.

Customers of MariaDB Corporation Ab include well-known companies such as Nokia, Red Hat, Samsung, ServiceNow, Walgreens and so on. (MariaDB Corporation Ab. 2020). One of the additional enterprises features the MariaDB corporation offers is MariaDB Enterprise Server which is an enhanced, hardened and secured version of a MariaDB Community Server (MariaDB Corporation Ab. 2020).

1.3 Objectives and scope

Though this is a product-oriented thesis, it will not include all development processes related to graphical user interface such as prototype designs, user experience designs and user story due to confidential information of the company. However, information related to MaxScale can be exposed as it is an open source project, part of graphical user interface' development processes can be described in thesis. As so, the primary objectives of this thesis can be divided into three categories as follows:

- Facilitating developer's environment which includes the setting up of MaxScale environment, Vue.js' development environment using VS Code.
- Improvement of MaxScale REST API for using in web application in terms of MaxScale user authentication. The application should prevent some common security issues in web application such as MITM, XSS, CSRF. This is because the REST API was developed and optimized for using in web application but CLI tools called maxctrl.
- Implementation of graphical interface for user authentication.

1.4 Support from the commissioning party and copyrights

Throughout the process of building the application, MaxScale development team at MariaDB Corporation Ab will make any changes to the REST API when necessary. The thesis writer will implement the GUI by using Vue.js framework along with consulting and observing ready-made application using similar technology stack at MariaDB Corporation Ab.

MariaDB Corporation Ab holds all rights to the MaxScale GUI application, including the product described in this thesis. A thesis commissioning agreement and confidential agreement were signed by the commissioning party, thesis advisor, thesis evaluator and me.

2 Theoretical framework

This chapter divides into three sections in order to provide technology knowledge to set up development environment and tools for developing the administrative user interface of MaxScale. In 2.1, this section gives an introduction about the open source Javascript framework Vue.js as well as its related support plugins and libraries. The discussion and explanation in this section covers the comparison of Vue.js to React.js to some extent. In 2.2, it provides an overview about MariaDB platform which focusing on the MaxScale product and its usage scenarios around MariaDB Server relational database. Finally, in 2.3, this section describes the usages of Docker and explains why it is handy for facilitating MaxScale development environment.

In general, this chapter will not intend to cover all knowledge related to Vue.js but the most important parts of the framework. Other parts related to the implementation of the UI will be discussed in the empirical part.

2.1 Vue.js

Vue.js is an opensource Javascript framework that claims to be the first and the only progressive framework until now. It was created in July 2013 by Evan You who was working at Google Creative Lab. His initially intention was to make an innovation of Angular Javascript framework to develop prototype faster. Therefore, his personal project has similar characteristics, outstanding features of Angular but more concise and easier syntax.

In the following years, the framework is redeveloped by taking a lot of ideas from other famous frameworks, libraries which includes React, Angular and so on. The framework has been quickly adapted, supported by the developer community and become a “progressive” framework. (Vue NYC 2017)

A progressive framework in terms of Vue.js, meaning that Vue.js is just a view layer focused framework but having opt-in official libraries supported. As it “is designed from the ground up to be incrementally adoptable”. Depending on the business needs, its support libraries, plugins are needed while reserving ability to integrate with other libraries or existing projects. Vue.js community has developed many opt-in libraries such as Vue Router (router library for creating single page application), Vuex (state management library), Vue Server Renderer (server-side rendering library). With its well adopted ecosystem libraries and plugins, Vue.js has ability to build single page application, progressive application, desktop application or even mobile applications. Standing in the market among other frameworks and libraries backed by big companies and large teams,

Vue.js brings the upsides but also eliminates the downsides of those frameworks and libraries (Vue.js 2020).

As for this reason, MariaDB Corporation Ab chose Vue.js to be the framework that will build the administration user interface of MaxScale.

Vue.js and its ecosystem is growing faster and faster in these recent years, as it is proved to be the front-end framework that has the most stars in GitHub in 2019 (bestofjs 2019). Since the framework itself is in the fast-growing track, to build high-performance and scalable web applications, the needs for understanding the keys concepts and its ecosystem of Vue.js is inevitable. The indispensable key knowledge of the Vue.js which need to be digested when building a Vue.js application is Reactivity system in Vue.js and the Vue instance lifecycle hooks.

Besides, to make a single page application scalable, reusable and well configured, Vue.js libraries like Vuex state management, Vue Router and Vue.js Tooling Vue CLI are recommended. For the case of building MaxScale GUI, MariaDB follows Material UI design concept, therefore, Vuetify, a Vue UI components library is chosen.

2.1.1 Reactivity system in Vue.js

The needs of a reactivity system come from the edge case of a typical application which is developed based on components approach and used Virtual DOM. Think of any applications as a tree component, there are parent component and its children components.

For example, with React library, although, it does track on the dependencies and update the DOM whenever there are changes. The child components do not automatically know to not re-render when the parent component's state or props changes. Nevertheless, with the use of an API called "shouldComponentUpdate" or converting stateful component into PureComponent in React, the issue can be solved (React 2020, JAVASCRIPT REPORT 2017). As so, the React library is not fully reactivity, it needs the manual process of tracking.

A reactivity system should ensure any updates on the parent component will not trigger unnecessary re-render on the child components, otherwise, large application has tremendous child components may encounter performance issue.

While other Javascript frameworks and libraries requiring the implementation of reactivity system or implement observer patterns, method to achieve reactivity. Vue.js implants its

reactivity system in every vue component instance so the developers have an optimized reactivity system out of the box. Figure 2 is the diagram which demonstrates the reactivity system in every vue components.

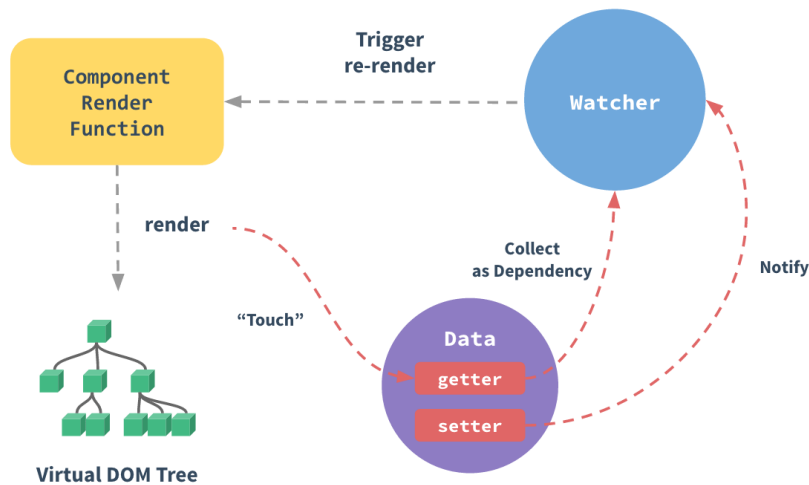


Figure 1. Diagram of reactivity system in Vue.js (Vue.js 2020)

Reactivity system in Vue.js can be seen as a system that has an observation system called the “Watcher”, a data object, a Component Render Function that interactive with a Virtual DOM Tree. The process of the reactivity system in Vue.js is can be understood as follows.

First, when the Vue instance is created, all the properties in the data object is passed to the Vue instance. It is then added to Vue’s reactivity system and converted to getter/setter methods using Object.defineProperty static method. Next, the Watcher collects the dependencies by calling the getter function which is defined in the Object.defineProperty method. After the first render, whenever there are changes (“Touch”) from the data object, the setter will be called to update the value in the data object while calling the Notify function. By doing this, the Watcher will trigger the re-render of the component (Vue.js 2020).

However, not all changes will be detected by Vue’s reactivity system including the deletion, addition of object properties and the changes of the element’s value based on the index of an array or the modification of the length in the array.

For example, with object modification, as it is illustrated in figure 2, the data object will be passed to the reactivity system and when adding a property named b, this will not trigger re-render. This is because when the Vue instance is created, all of object properties must be passed to the reactivity system in order to be tracked and become reactivity.

```
var vm = new Vue({
  data: {
    a: 1
  }
})
// `vm.a` is now reactive

vm.b = 2
// `vm.b` is NOT reactive
```

Figure 2. Data object in vue instance (Vue.js 2020)

To handle this limitation of Vue.js, Vue provides a set method from Vue instance to update nested object as shown in figure 3A.

```
<body>
<div id="app">
  <p>{{todos}}</p>
</div>
<script>
var vm = new Vue({
  el: "#app",
  data: {
    todos: ["writing theoretical part", "writing empirical part"]
  },
  created() {
    // This is reactive
    Vue.set(this.todos, 0, "Writing introduction first");
    // Or using splice method from Array prototype
    // this.todos.splice(0, 1, 'Writing introduction first')

    // Alternative, using alias instance method $set
    //this.$set(this.todos, 0, "Writing introduction first");

    // Using splice to change the length of the array
    this.todos.splice(1); // this change the length from 2 to 1.
  }
});
</script>
</body>
```

A

```
<body>
<div id="app">
  <p>{{person}}</p>
</div>
<script>
var vm = new Vue({
  el: "#app",
  data: {
    person: {
      eyeColor: "blue"
    }
  },
  created() {
    // This is reactive
    Vue.set(this.person, "hairColor", "yellow");
    // Alternative, using alias instance method $set
    // this.$set(this.person, 'hairColor', 'yellow')
  }
});
</script>
</body>
```

B

Figure 3. Object and Array changes detection

In terms of Array, we can also use the Vue set method to update the value of an element at a specific index, as it is illustrated in figure 3B.

2.1.2 Vue instance lifecycle hooks

Similar to other frameworks and libraries using Virtual DOM, Vue.js also has a lifecycle that goes through a series of steps when a vue instance is created as it is illustrated in figure 5. Throughout the process, Vue.js triggers eight functions called lifecycle hooks that can be overridden to run code at specific stages (Vue.js 2020).

Vue.js has almost the same lifecycle and allows user to add their own code at specific stages as React.js does. React.js has three main stages including Mounting, Updating and Unmounting as shown in figure 4. Meanwhile, Vue.js lifecycle can be divided into four main stages which includes Creation, Mounting, Updating and Destruction (Unmounting).

Vue.js just has one more stage called the “Creation” which is the very first stage of the lifecycle. There are two lifecycle hooks in this “Creation” stage including “beforeCreate” and “created”. In this stage, vue.js initializes the dependencies, events, lifecycle and adds data to reactivity system after the vue instance is created. Therefore, this is not the stage where user can manipulate the DOM because it’s not available yet. The “beforeCreate” hook is not used as often as the “created” hooks since the reactivity data and events are not accessible. With the “created” hooks, user can perform changes to the reactivity data before the data is rendered in the DOM or even handle server-side rendering effect. For example, the application needs to sort the data in the table before sending and rendering it in the client browser.

The “Mounting” stage in Vue.js is similar to React.js as both allow user to interact and work with the DOM. However, the “Mounting” stage in React.js offers only one lifecycle method called “componentDidMount” while Vue.js provides two lifecycle hook methods including “beforeMount” and “mounted”. Although, in most usage scenarios, the “beforeMount” hook is not the most used as the mounted hook. This is because of few reasons as follows:

- The “beforeMount” hook is called just right after the “created” hook. But it does not support sever-side rendering. Hence, the “created” hook is more favorable.
- User cannot perform DOM manipulation since this hook is called before the mounting stage when the replacement of Virtual DOM with DOM is not done yet. The “mounted” hook is more useful since the mounting stage has been done.

The “Updating” stage in Vue.js occurs when there are changes on the reactivity data. For example, when user clicks the button, the reactivity data controls the visibility of the modal

will be changed. Vue.js is then compute the data in the Virtual DOM to re-render and patch the update to the actual DOM.

In this stage, it has two lifecycle hooks consists of “beforeUpdate” and “updated”, these seems to be useful when user wants to detect changes on the DOM, perform DOM manipulation, access to reactivity data. However, to prevent unnecessary updates on the DOM, changing reactivity state in these hooks is not recommended by Vue.js (Vue.js 2020).

Unlike Vue.js, the “Updating” stage in React.js provides more methods comprise “getDesiredStateFromProps”, “shouldComponentUpdate”, “getSnapshotBeforeUpdate” and “componentDidUpdate”. The usages of these methods are not similar to Vue.js updating hooks as it focuses more on the ability to control the reactivity of the data.

Finally, regarding the Destruction or Unmount stage, both React.js and Vue.js invokes this stage to destroyed and removed the component from the DOM to prevent memory leak in the application. React.js provides “componentWillUnmount” method to perform cleaning up effect, cancelling event listener or network requests. Vue.js also allows users to do the same actions with “beforeDestroy” lifecycle hook. Besides, Vue.js provides one more hook called “destroyed” which triggering after everything is cleaned up. This additional hook compares to React.js seems to be redundant in terms of normal application cases, though it may be used to do last task like informing the server that the component is destroyed. (alligator.io 2017).

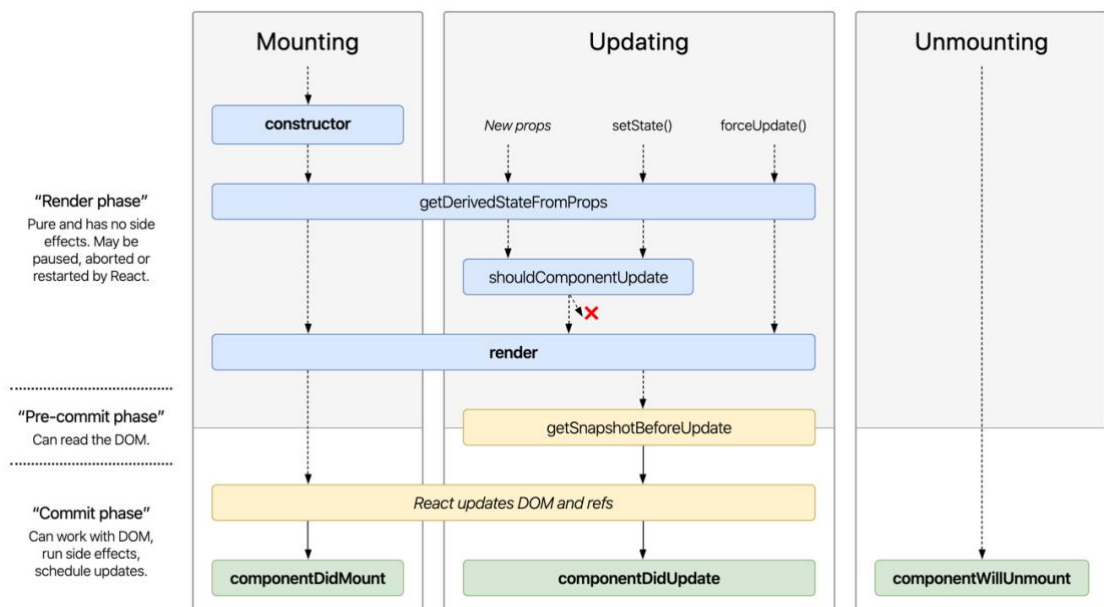
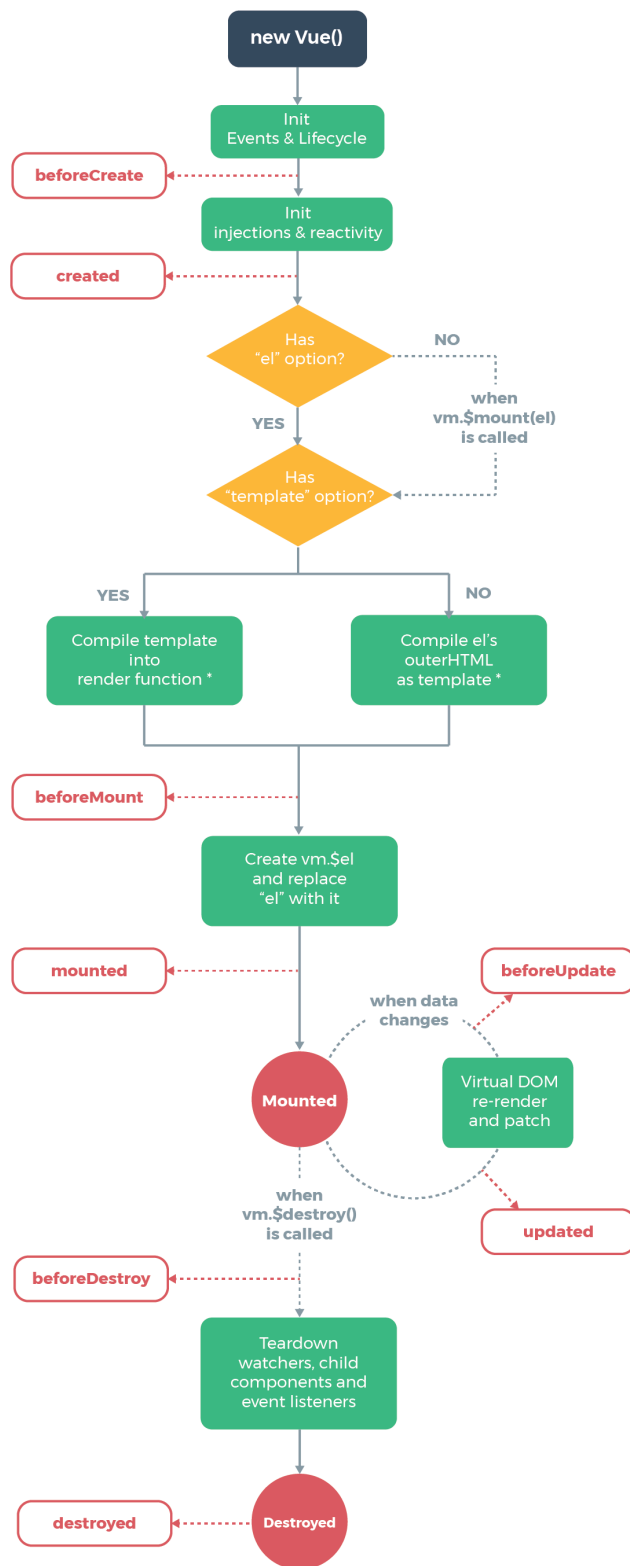


Figure 4. React.js Lifecycle diagram (Reactjs.org 2020)



* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

Figure 5. Vue Lifecycle diagram (Vue.js 2020)

2.2 MariaDB Platform

This section provides sufficient knowledge relates to the MariaDB platform along with explanations of specific related terminologies such as database server, database proxy and route splitting (MariaDB Corporation Ab 2020). Figure 6 illustrates MariaDB Platform X4 includes MariaDB Enterprise Server version which is required subscription. However, MaxScale is not limited to be used with MariaDB Enterprise Server version, it can also be used within the open source MariaDB server with a proprietary license. It means MaxScale is free if it is used with less than three MariaDB servers (MariaDB Foundation 2016).

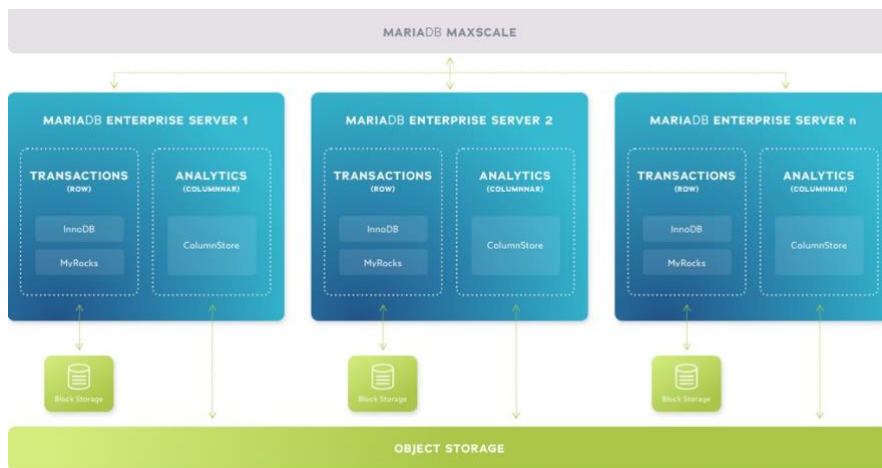


Figure 6. MariaDB Platform X4 (MariaDB Corporation Ab 2020)

2.2.1 MariaDB Server relational database

MariaDB Server relational database is a well-known open source database server that has been guaranteed to be open source and backed by the MariaDB foundation community as well as MariaDB Corporation Ab company. Most Linux distributions and cloud service providers have MariaDB Server installed due to its performance, stability, and open source (MariaDB Foundation 2020). MariaDB Server relational database named its database with the word “server” as it operates as a typical database server.

In simple terms, a database server is a data warehouse used to store websites, data, information and make the databases available to the internet users. A database server consists of a database management system (DMBS) and database with many core functions such as recovery services, query management system and security measurement service. Depends on the client requests, database server searches in

databases for specific records and send over the network to the client. In other words, a database server can be considered as a server that provides database services or a server that runs database systems. (Science Direct 2020).

For MariaDB Server, it is created to host multiple SQL databases in the same machine and manage any traffics between the client application and the relational databases. MariaDB server has many outstanding features such as speed, scalable and powerful in handling large data. In addition, MariaDB Server is under General Public License v2.0 (GPLv2) forever which means that anyone can have access to the source code. Therefore, it ensures the transparency, security, availability of the database (MariaDB Corporation Ab 2020; MariaDB Foundation 2016; MariaDB Foundation GitHub 2019)

2.2.2 MariaDB MaxScale

The traditional application usually consists of two layers which are the application itself and the primary server layer that connects together. The primary server layer of MariaDB platform is the MariaDB server which hosts the relational databases as mentioned in [2.2.1](#) section. When the application sends request directly to the server layer, the server layer searches in the databases for the request record and sends it to the application (figure 7A). However, when the application grows or the number of requests from the users increases and a single primary server is unable to process the workload or maybe even worse the primary server can encounter technical failures. Inevitably, the primary server needs to be replaced by a replica of that primary server. Nevertheless, this requires manual, repetitive work from develop operation team to tackle that situation. It would be a waste of time and labor if a large and complex application requires several replicas of the primary server confronting that incident (figure 7B).

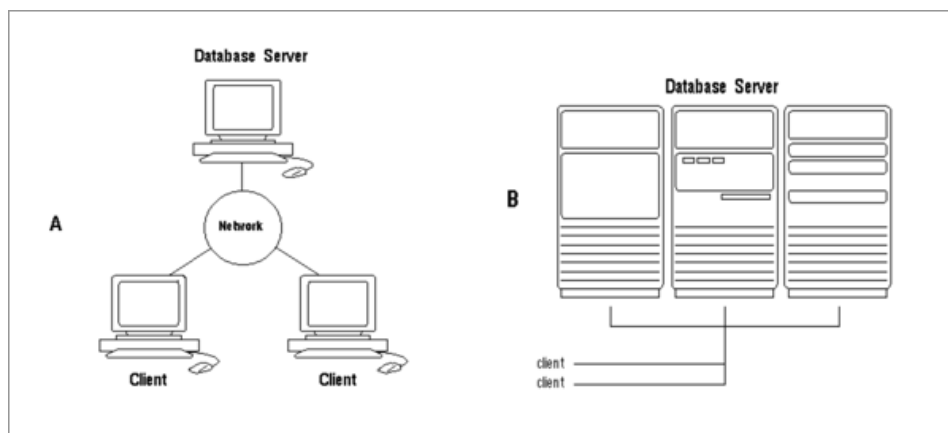


Figure 7. The Client/Server Architecture and Distributed Processing (Oracle 1999)

As for the need of the automation work, MaxScale is an automated service that is created in order to extend “the high availability, scalability and security of MariaDB Server” (MariaDB Corporation Ab 2020) relational database. It is a database proxy that support the development of the application in terms of database server infrastructure. A database proxy can be described as a middle layer that locates between the database server and the application (figure 8).

Since the application is connected to MaxScale database proxy, any traffic data requests from the client application will be sent to MaxScale which be eventually forwarded to the actual database server. MaxScale database proxy will constantly perform health check on the primary server whenever there are requests from client application to ensure the server is available to access. As the result, MaxScale will automatically carry failover which is a method to prevent the database server from failure by immediately replacing the primary server with its replicas. Approximately the same time, the previous transactions which is sent to the failed primary server will be forwarded to the new primary database server. By performing this failover method and other unmentioned methods, MaxScale ensures the high availability of the application to the user.

MaxScale not only ensures the availability and hardens the security by supporting database firewall but also supports scalability of the database by utilizing its “built-in plugins for multiple routers, filters and protocols” (MariaDB Corporation Ab 2020). For example, a travel fare metasearch engine application certainly has more read requests from the users more than the write requests. By setting up the “Read/Write Splitting” service (MariaDB Corporation Ab 2020), MaxScale will forward any write transactions to the primary sever while read transactions are forwarded to the replicas servers which is also known as slave servers. As MariaDB Corporation states “MariaDB MaxScale can be configured to forward database requests and modify database responses based on business and technical requirements” (MariaDB 2019; MariaDB Corporation Ab 2020; Severalnines 2018).

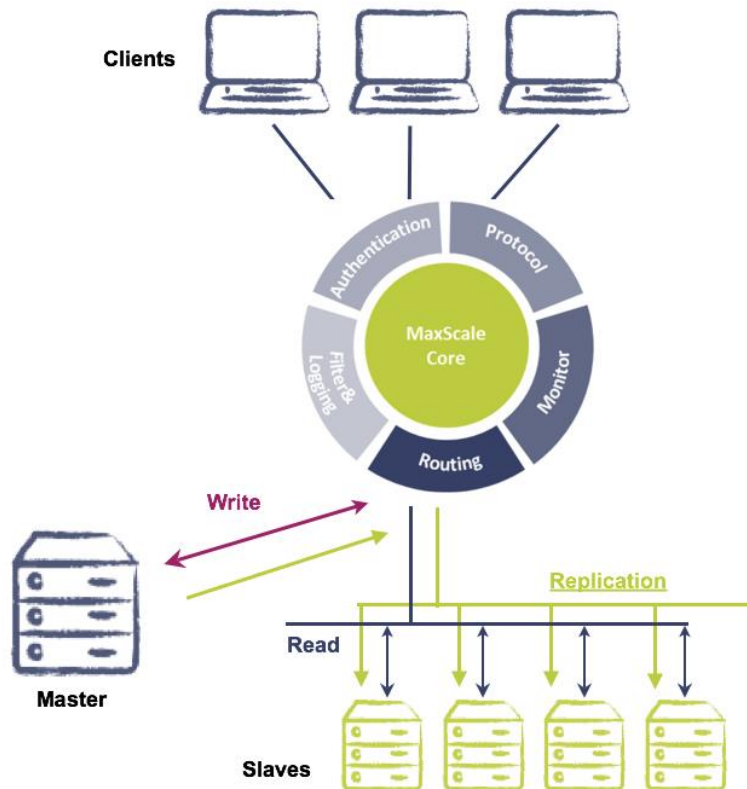


Figure 8. MariaDB MaxScale (MariaDB Corporation Ab 2017)

As for the installation of MaxScale, MariaDB corporation Ab provides several package installations includes deb, rpm and Tarball which are archive files format for Linux distributions only (StackExchange UNIX&LINUX 2013).

In addition, MaxScale's source code is available to be accessed in GitHub, therefore MaxScale can be built from the source code with specific versions (MariaDB Corporation Ab 2020, Thien, L. 2020).

2.3 Docker

As mentioned in 2.2.2, MaxScale is used when there are multiple MariaDB servers running. In practice, there will be several physical servers that each of server running MariaDB server instance. Alternatives, many cloud service providers such as Amazon Web Service, Google Cloud and Alibaba Cloud provide service for hosting MariaDB servers (Amazon Web Services 2020; Google Cloud Platform 2020; Alibaba Cloud 2020). However, for the purpose of developing the admin dashboard user interface for MaxScale, simulation of several MariaDB server instances is sufficient.

At the moment, there are multiple ways to run multiple MariaDB server instances in the same machine includes virtual machine, container or following "Configuring Multiple MariaDB Server Processes" tutorial from MariaDB corporation Ab (MariaDB Corporation

Ab 2020). For this thesis project, using Container is the most convenient approach due to the following reasons:

- For developing the GUI of MaxScale, an end to end application stack needs to be set up and run in the same host computer which includes multiple MariaDB server instances, MaxScale and the front-end local hosting environment. However, setting up and running this application stack by using VM software, it will consume system resources from the host computer significantly. In addition, setting up the environment using VM requires time and a lot of configurations (BackBlaze 2018).
- Following the tutorial from MariaDB Corporation is complex and the configuration depends on the operating system of the host computer.
- Container and VM may have the same usage purpose which is to run applications and software in multiple operating systems, but its design and approach are different. VM emulates computer system that “virtualizing the underlying computer” while Container only virtualizes the operating system As it is illustrated in figure 9, there are no Guest Operating System layers compared to VM, so instead of reproducing the same operating system code, creating independent Guest OS layers, Container shares the operating system resources between operating systems. This makes Container faster and lighter compared to VM.
- For the sake of simplicity and flexibility, Container is better option as it allows to rapidly change the configuration file effortlessly while VM requires to reinstall and setting up most of the things.

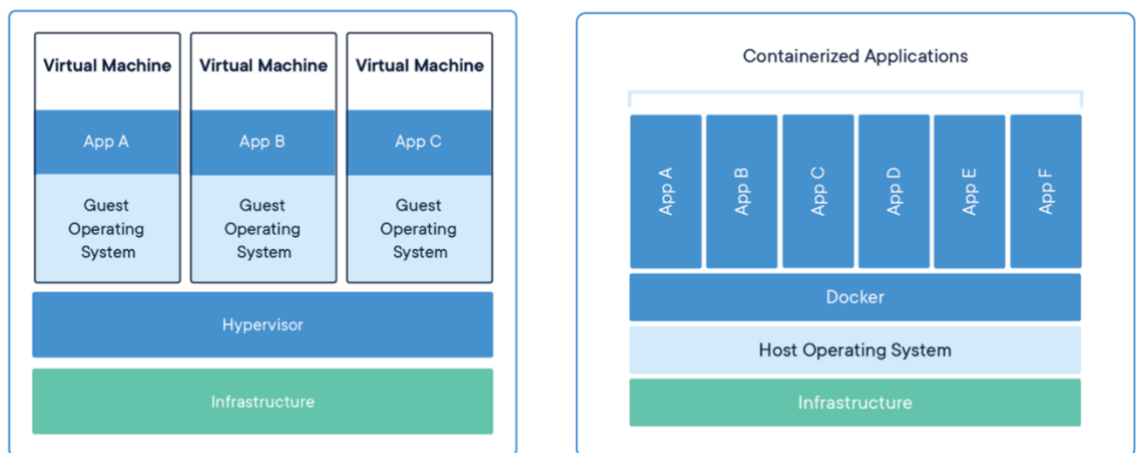


Figure 9. VM architecture and Container architecture (Docker 2020)

Docker container is chosen for this thesis project as the thesis writer already has experience with it and MariaDB Corporation also support the use of Docker for their MariaDB server. MariaDB Server docker image is already deployed and maintained by

MariaDB Corporation Ab (Docker Hub 2020). However, there are two version of MariaDB server image in Docker Hub which are mariadb/server and mariadb image. Usage of either mariadb/server image or mariadb image is compatible with MaxScale. The different is that mariadb/server image is maintained by MariaDB Corporation Ab while mariadb image is developed and maintained by the Docker Community (Docker Hub 2020).

3 Empirical

As mentioned in the introduction, the objective of this thesis is to set up developer's working environment to be eventually used for implementing the administrative UI of MaxScale. The UI will be built by using Vue.js framework as a user interface framework and the following fundamental Vue.js libraries such as Vuetify, a reusable components library followed material design concept, Vuex as a state management library and a starter kit called Vue CLI using to automatically generate project boilerplate with opt-in and opt-out configurations.

The empirical chapter will be divided into three main sections including: Initializing the project, MaxScale REST API authentication approach and Implement the graphical user interface of authentication page. The first section focuses on setting up tools, development environment in terms of MaxScale and Vue.js. The second section discusses around current authentication method in MaxScale REST API and new authentication approach for using in web application. The last section presents UI implementation of the authentication page.

Generally, this chapter walks through the most vital aspects of setting up the development environment and a part of building the application by following good coding practices and modular approaches.

3.1 Initializing the project

3.1.1 Setting up MaxScale environment

As mentioned in [2.2.2](#), MaxScale can be installed to the host OS by several approaches, either by using released pre-built packages or by building MaxScale from the GitHub source code. Since the development of the GUI for MaxScale is still ongoing, changes on the MaxScale source code is required, therefore building MaxScale from the source code is inevitably required. This approach allows users to build their own binaries of MaxScale whether it is the released version or development version. However, the installation steps

may be tricky and time consuming than using pre-build packages as discussed in the following content.

Regarding the operating systems that MaxScale supports, MaxScale can be built in any systems having the required main core packages (Table 1). Since, most of these packages are developed on top of Linux distributions, hence the host operating system is recommended to build MaxScale is Linux.

Table 1. Required installed packages (MariaDB Corporation Ab 2020).

Package	Version
CMake	2.8.12 or later
GCC	4.4.7 or later
SQLite3	3.3 or later
OpenSSL	Not specified
Bison	2.7 or later
Flex	2.5.25 or later
libuuid	Not specified
GNUTLS	Not specified
libcurl	Not specified

The thesis writer is using elementary operating system, which is a Linux distribution based on ubuntu, therefore, those mentioned packages can be installed without difficulty. Besides those packages, MaxScale installs several packages under the hood by running a shell script calls “install_build_deps.sh” as illustrated in figure 10.

```
git clone https://github.com/mariadb-corporation/MaxScale
mkdir build
cd build
../MaxScale/BUILD/install_build_deps.sh
cmake ../MaxScale -DCMAKE_INSTALL_PREFIX=/usr
make
sudo make install
sudo ./postinst
```

Figure 10. MaxScale building from source code build steps (MariaDB Corporation Ab 2020)

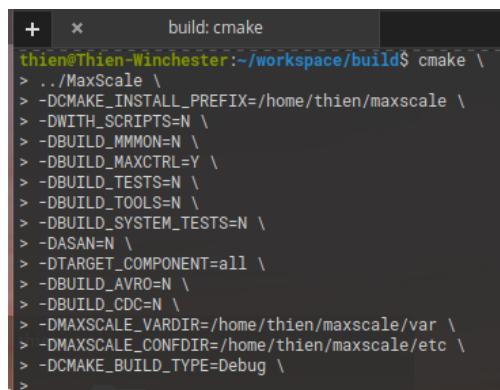
MaxScale development process including bug fixes, enhancements patches, new features occurring every working days. Apart from that, the development of the REST API partially depends on the requirements needed from the GUI. As part of development process,

every time MaxScale repository has changes, MaxScale needs to be built and installed to the host OS again. Nevertheless, the GUI does not need all features from MaxScale such as system tests, build but it certainly needs REST-API feature, maxctrl to be enabled. Fortunately, MaxScale allows to configure the build options by using cmake command, all available options can be found on MaxScale GitHub.

From MaxScale GitHub, before building MaxScale, the document instructs to execute the following command as shown in figure 10:

```
cmake ../MaxScale -DCMAKE_INSTALL_PREFIX=/usr
```

-DCMAKE_INSTALL_PREFIX is a built option indicating the target folder MaxScale will be installed in which allows multiple versions of MaxScale to be installed in the same OS. As mentioned, the GUI will not need all features of MaxScale, figure 11 shows a list of build options to instruct MaxScale to have a minimum build version by opting out unnecessary parts. After configuring the build options, MaxScale can be built by simply run the “make” command.



```
thien@Thien-Winchester:~/workspace/build$ cmake \  
> ../MaxScale \  
> -DCMAKE_INSTALL_PREFIX=/home/thien/maxscale \  
> -DWITH_SCRIPTS=N \  
> -DBUILD_MMMON=N \  
> -DBUILD_MAXCTRL=Y \  
> -DBUILD_TESTS=N \  
> -DBUILD_TOOLS=N \  
> -DBUILD_SYSTEM_TESTS=N \  
> -DASAN=N \  
> -DTARGET_COMPONENT=all \  
> -DBUILD_AVRO=N \  
> -DBUILD_CDC=N \  
> -DMAXSCALE_VARDIR=/home/thien/maxscale/var \  
> -DMAXSCALE_CONFDIR=/home/thien/maxscale/etc \  
> -DCMAKE_BUILD_TYPE=Debug \  
>
```

Figure 11. MaxScale build options for minimum build version.

When Maxscale finishes its build process, executing “sudo make install” will install MaxScale using the build version which was previously built. However, there is one thing to considered whether MaxScale needs to be run with root privileges or not. If MaxScale is installed with sudo command, “/maxscale” folder and all of its files can only be executed by the root owner. However, even if MaxScale is run with sudo command, an alert will be printed said “MaxScale cannot be run as root. This apparently a typo in the documentation, so instead of using “sudo make install”, the correct command will be “make install”.

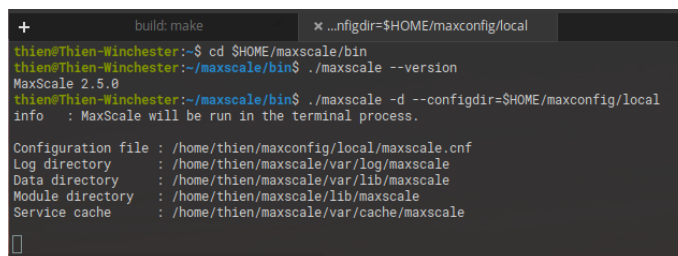
By default, MaxScale reads configuration file with the “.cnf” extension in the “etc” directory at this path maxscale/etc” (MariaDB Corporation 2020). However, for the purpose of developing, several build versions may use the same configuration file. As so, new path for the configuration file will be defined before starting MaxScale. Meaning that, instead of reading configuration files the “etc” directory, it reads files from specified path.

For example, if the configuration file is located at this path “/ \$HOME/maxconfig/local”.

In order to run MaxScale, the current working directory has to be changed to this path “/ \$HOME/maxscale/bin”, then executing the following command:

```
./maxscale -d --configdir=$HOME/maxconfig/local
```

MaxScale will be run using the specified configuration file as illustrated in figure 12.



```
thien@Thien-Winchester:~$ cd $HOME/maxscale/bin
thien@Thien-Winchester:~/maxscale/bin$ ./maxscale --version
MaxScale 2.5.0
thien@Thien-Winchester:~/maxscale/bin$ ./maxscale -d --configdir=$HOME/maxconfig/local
info : MaxScale will be run in the terminal process.

Configuration file : /home/thien/maxconfig/local/maxscale.cnf
Log directory      : /home/thien/maxscale/var/log/maxscale
Data directory     : /home/thien/maxscale/var/lib/maxscale
Module directory   : /home/thien/maxscale/lib/maxscale
Service cache      : /home/thien/maxscale/var/cache/maxscale
```

Figure 12. Running MaxScale.

For developing purpose, the basic configuration for MaxScale needs to define two servers with id: row_server_1 and row_server_2 with their address, port and protocol as shown in figure 13. Though, the figure shows only partial configuration which includes server, listener and service.



```
GNU nano 2.0.6 File: maxscale.cnf
type=service
router=readconnroute
router_options = master
servers = row_server_1,row_server_2
user = maxskysql
password = skysql

[RCR-Writer-Listener]
type = listener
service = RCR-Writer
protocol = mariadbclient
port = 3308

[row_server_1]
type = server
address = 127.0.0.1
port = 4001
protocol = MariaDBBackend

[row_server_2]
type = server
address = 127.0.0.1
port = 4002
protocol = MariaDBBackend
```

Figure 13. Basic MaxScale configuration file

These servers will be run in docker containers using docker-compose tool which allows to create and run multiple containers together in an isolated environment. (Figure 14) (Docker 2020).

```
version: "3.3"
services:
  row_server_1:
    image: mariadb/server:10.4
    network_mode: "host"
    environment:
      MYSQL_ROOT_PASSWORD: mariadb
    volumes:
      - ./sql/master:/docker-entrypoint-initdb.d
    command: mysqld --log-bin=binlog --binlog-format=ROW --server-id=1000 --port=4001 --log-slave-updates

  row_server_2:
    image: mariadb/server:10.4
    network_mode: "host"
    environment:
      MYSQL_ROOT_PASSWORD: mariadb
    volumes:
      - ./sql/slave:/docker-entrypoint-initdb.d
    command: mysqld --log-bin=binlog --binlog-format=ROW --server-id=1001 --port=4002 --log-slave-updates
```

Figure 14. docker-compose.yml file.

In the docker.compose.yml file, the volumes section allows to persist and share data between containers. In this case, there is a “/sql” directory containing two child directories master and slave. “:/docker-entrypoint-inidb.d” simply executing the script inside master and slave directory. Script in the master directory creates a test database and users as a primary server while the script in the slave directory sets up the replication mechanism for the primary server (figure 15).

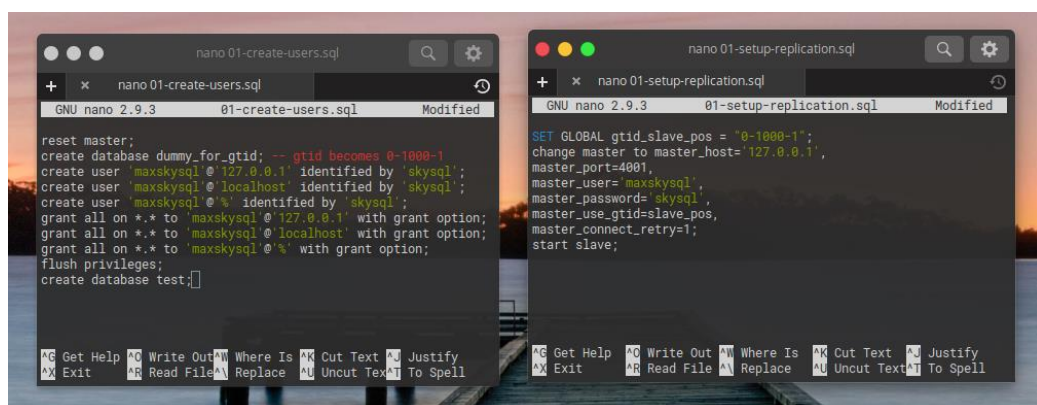


Figure 15. Persisting and sharing data between primary server and slave server

After defining docker-compose.yml file, executing docker-compose up will create isolated container for each server and run those servers. When all servers are running, the status of the servers can be checked through maxctrl CLI with the command “maxctrl list servers”. If the state of the row_server_1 is “Master, Running” and the row_server_2 is

“Slave, Running”, the configuration of MaxScale environment is done properly as expected (figure 16).

```
thien@Thien-Winchester:~$ maxctrl list servers
```

Server	Address	Port	Connections	State	GTID
row_server_1	127.0.0.1	4001	0	Master, Running	0-1000-9
row_server_2	127.0.0.1	4002	0	Slave, Running	0-1000-9

```
thien@Thien-Winchester:~$
```

Figure 16. Output of the “maxctrl list servers” command.

3.1.2 Vue CLI and Visual Studio Code configuration

The GUI for controlling MaxScale will be bootstrapped by Vue CLI which is an official tool from Vue.js for creating project boilerplate with opt-in and opt-out features either by using CLI or GUI approaches.

The command for creating the project by using CLI approach is “vue create maxgui”. When creating the project with this approach, user has ability to manually select needed features. The initial features for this project consist of Babel, Vue Router, Vuex, CSS Pre-processors (Sass/SCSS with dart-sass), Linter / Formatter, Unit Testing and E2E Testing. These features are carefully considered to be added in this project in order to remain the consistence and convention among other MariaDB Corporation Ab user interface projects using Vue.js.

The structure of the project will have babel and eslint configurations located in their dedicated configuration files to facilitate the ease of later configuration. However, this is minimum configurations for starter project, it is not enough for a real-world application that requires to be scalable. Fortunately, Vue CLI provides an optional dedicated file named “vue.config.js” in the root project folder to configurate its features. By configuring the vue.config.js file, we can handle how webpack process to bundle our application (figure 17).

```

const path = require('path')
process.env.VUE_APP_VERSION = require('./package.json').version

module.exports = {
  chainWebpack: config => {
    const types = ['vue-modules', 'vue', 'normal-modules', 'normal']
    types.forEach(type => addStyleResource(config.module.rule('scss').oneOf(type)))
  },
  configureWebpack: {
    resolve: {
      modules: [path.resolve('./src'), path.resolve('./node_modules')],
    },
    devServer: {
      progress: false,
      port: 8000,
      headers: {
        'Access-Control-Allow-Origin': '*',
      },
      proxy: {
        '/': {
          changeOrigin: true,
          target: process.env.VUE_APP_API,
        },
      },
    },
  },
  transpileDependencies: ['vuetify'],
  outputDir: `${process.env.buildPath}/gui`,
  pluginOptions: {
    i18n: {
      locale: 'en',
      fallbackLocale: 'en',
      localeDir: 'locales',
      enableInSFC: true,
    },
  },
  productionSourceMap: false,
}

function addStyleResource(rule) {
  rule.use('style-resource')
    .loader('style-resources-loader')
    .options({
      patterns: [path.resolve(__dirname, './src/styles/constants.scss')],
    })
}

```

Figure 17. vue.config.js

The very first thing that needs to be configured regardless of production application or development application is handling CORS (Cross-Origin Resource Sharing). This is a common task when developing the user interface of any applications requires the communication between the backend and the front-end. For production application, this requires the configuration from the web server, however, when the development application is served locally on the host machine, a proxy needs to be configured in the development server to bypass CORS.

Secondly, when using Sass Pre-processor, using variables or constants sass/scss files within vue template is a typical development approach. Instead of importing needed styles files every time to vue template component, we can configure to automatically import necessary style files to the target vue template component.

Thirdly, for UI frameworks or libraries similar to Vue.js that developed based heavily on reusable component approach, user usually has to import child components to parent

component. To facilitate development effortlessly, we need to configure webpack to resolve component path to module path.

```
configureWebpack: {  
  resolve: {  
    modules: [path.resolve('./src'), path.resolve('./node_modules')],  
  },  
},
```

Figure 18. Resolve path to use module path.

For example, with the webpack configuration as shown in figure 18. To import “Bar” component in “src/components/common/Bar.vue” and a plugin “moment” from “node_modules” to a Foo component located at this path “src/pages/Foo.vue”, we can simply import components with module path as follows:

```
import Bar from 'components/common/Bar'  
import moment from 'moment'
```

Other options such as “transpileDependencies”, “outputDir”, “pluginOptions”, “productionSourceMap” are highly recommended to configure.

- The “transpileDependencies” option is used to target specific packages in “node_modules” to be transpiled alongside with the application when using Babel. Since, this project will be bootstrapped by using Vuetify, it is advised to transpile this library.
- The “outputDir” should be added to specify the output directory when the application is built. Because the GUI will be served by MaxScale in MaxScale’s share folder, therefore, in my host machine, the “outputDir” value will be as follows “/home/thien/maxscale/share/maxscale/gui”.
- The “pluginOptions” should be added when using external library like “vue-i18n” which is a plugin to have internationalization application.
- If the “productionSourceMap” option is set to be true, it simply informs webpack to create source map files that enable browser debugging tool to map the transpiled source code to the original source code.

Apart from that, Vue CLI allows to create node environment variables to improve and optimize product development workflow by creating “env.local” (local environment variable), “env.development” (development environment variable) or “env.production” (production environment variable).

The local environment variable is created with an intention to be ignored by git version-control system and used for assigning local variables in the host system. In the case of MaxScale, for outputting the built directory to MaxScale, the absolute path to MaxScale share folder needs to be specified. Since the location of MaxScale share folder is different for each user, a local environment variable that assigning MaxScale share directory path should be created before building the application. This can be assigned as follows:

```
"buildPath=/home/thien/maxscale/share/maxscale/gui"
```

Therefore, instead of assigning outputDir to the actual MaxScale share directory, we assign to it with the local environment variable in this way: outputDir:

```
`${process.env.buildPath}/gui`
```

Basically, when the application is built or served, the "env" object variable will be created as a property of the global "process" object created by Node.js. As so, the application has access to all existing environment variables from the global "process" object.

(Vue CLI 2020, TWILIO INC 2017).

Browser compatibility is considered to a significant factor in benchmarking a web application. When creating the project with Vue CLI, a ".browserslistrc" file contains a value "defaults" will be created automatically. Vue CLI will use the value in this file to transpile needed JavaScript features and add needed css vendor prefixes. Depending on business need, the value of this file will be varied while keeping the cross-browser compatibility to some extent (Vue CLI 2020).

Regarding Visual Studio Code configuration, sharing settings, configurations and extensions to remain then consistency of the source code is an ideal choice for an application developed by several developers. In addition, this relieves the process of integrating one application to another application since the programming style is shared between developers and applications. Once the developer installs recommended VS code extensions, all settings in the "settings.json" file are automatically recognized by VS code as workspace configurations. The final configurations consists of list of recommended install extensions, workspace VS code "settings.json", configuration file of Prettier extensions using for formatting source code and a jsconfig.json file used to utilized VS code IntelliSense features (figure 19)

```

() extensions.json ×
.vscod > {} extensions.json > [] recommendations
1 {
2   "recommendations": [
3     "dbaeumer.vscod-eslint",
4     "esbenp.prettier-vscode",
5     "wix.vscod-import-cost",
6     "octref.vetur",
7     "eg2.vscod-npm-script",
8     "christian-kohler.npm-intellisense",
9     "streetsidesoftware.code-spell-checker",
10    "wayou.vscod-todo-highlight"
11  ]
12 }

() jsconfig.json ×
() jsconfig.json > {} compilerOptions > {} paths > [] *
1 {
2   "compilerOptions": {
3     "baseUrl": ".",
4     "allowSyntheticDefaultImports": false,
5     "paths": {
6       "*": ["src/*"]
7     }
8   },
9   "include": ["src/**/*"]
10 }

() .prettierrc ×
() .prettierrc > ...
1 {
2   "printWidth": 100,
3   "trailingComma": "es5",
4   "singleQuote": true,
5   "tabWidth": 4,
6   "semi": false
7 }

() settings.json ×
.vscod > {} settings.json > {} [javascriptreact]
1 {
2   "git.ignoreMissingGitWarning": true,
3   "javascript.updateImportsOnFileMove.enabled": "always",
4   "explorer.confirmDragAndDrop": false,
5   "explorer.confirmDelete": false,
6   "window.zoomLevel": 3,
7   "eslint.validate": ["javascript", "javascriptreact", "vue", "vue-
8     slint.alwaysShowStatus": true,
9     vetur.completion.tagCasing": "initial",
10    "[vue]": {
11      "editor.defaultFormatter": "esbenp.prettier-vscode"
12    },
13    "[javascript]": {
14      "editor.defaultFormatter": "esbenp.prettier-vscode"
15    },
16    "editor.multiCursorModifier": "ctrlCmd",
17    "[javascriptreact]": {
18      "editor.defaultFormatter": "esbenp.prettier-vscode"
19    },
20    "[html]": {
21      "editor.defaultFormatter": "vscod.html-language-features"
22    },
23    "prettier.trailingComma": "all",
24    "prettier.singleQuote": true,
25    "vetur.format.defaultFormatter.html": "prettier",
26    "prettier.printWidth": 100,
27    "[json]": {
28      "editor.defaultFormatter": "esbenp.prettier-vscode"
29    },
30    "sync.forceUpload": true,
31    "prettier.tabWidth": 4,
32    "prettier.useEditorConfig": false,
33    "vetur.format.options.tabSize": 4,
34    "prettier.semi": false,
35    "editor.formatOnSave": true,
36    "[jsonc]": {
37      "editor.defaultFormatter": "esbenp.prettier-vscode"
38    },
39    "cSpell.userWords": [
40      "camelcase",
41      "mariadb",
42      "vuetify"

```

Figure 19. Visual Studio Code configuration

3.1.3 Setting up necessary Vue.js plugins

Plugins in Vue.js can be thought as global features that are accessible throughout the application to speed up development progress. In addition, some of the plugins are used to build scalable, well organized and manageable application. Though, this project utilizes a few helpful plugins consisting of Axios, vue-fragment, portal-vue, vue-i18n, vue-moment, Vuetify and Vuex. The setting of most plugins is easily and even auto set-up when install the plugin through vue-cli-service of Vue CLI. Hence, this section focuses mainly on the setting up of Axios plugin.

A dashboard application certainly needs to send and retrieve data from the server by performing http requests. Back to the past, if a traditional application operates this process synchronously, it will freeze the current user interface of the application while making server calls which is bad in terms of user experience. Therefore, asynchronous development technique for Javascript was introduced in 1996 to solve synchronous issue in web application. Later on, other techniques were developed but Ajax developed by Google in 1999 is the most quickly adapted asynchronous development technique and

being stayed at the crown until now. Ajax techniques uses XMLHttpRequest internally to perform data exchanges between the servers and the client application without refreshing the browser (Wikipedia 2020).

Using Ajax techniques solves the synchronous issue in web application, but its syntax requires repetitive work. Whenever there is anything that is repetitive, there will be always a way to make it automation. As so, many libraries or APIs are developed on top of Ajax techniques to have better concise syntax. In term of using these libraries with Vue.js, Axios, an open source http client library is recommended to use in Vue.js because it has well-supported and maintained from large community. (Vue.js 2020)

Since, Vue.js allows to define new properties in the global “prototype” object, we can simply define axios as a new property of this object and is then available to access in all Vue instances. However, reinvent the wheel is not always ideal, the community of Vue.js provides a simple and lightweight wrapper plugin out of the box called vue-axios. This plugin internally uses the same approach by defining axios as a property to the Vue “prototype” object as shown in figure 20A. Figure 20B shows how effortlessly the usage of axios in Vue.js can be done.



```
Vue.axios = axios

Object.defineProperties(Vue.prototype, {
  axios: {
    get() {
      return axios
    }
  },
  $http: {
    get() {
      return axios
    }
  }
})
```

```
import Vue from 'vue' 68.1K (gz)
import axios from 'axios' 15.4K
import VueAxios from 'vue-axios'

Vue.use(VueAxios, axios)
```

A **B**

Figure 20. Code snippet of vue-axios and the usage of it within Vue.js.

As mentioned in the beginning of this section, other plugins are auto set-up when using Vue CLI except vue-fragment. This third-party plugin is added to this project as a workaround of the root element issue in vue template with the current Vue.js version (Vue.js GitHub 2017).

For example, with a single file component called “Hello.vue”, the Vue template requires one root element to wrapper the content inside the <template> tag. If the <template> tag has two child elements, the eslint validator will show the “vue/valid-template-root” error. With the help of vue-fragment plugin, the issue can be bypassed by adding a fragment tag <fragment></fragment> to wrapper the two child elements.

This fragment plugin is actually a borrowed concept from React.js version 16. In React.js, the syntax for the fragment is <React.fragment></React.fragment>. However, React.js provides an alternative and more concise syntax for the fragment which is just an empty tag: <></> (Reactjs.org 2020).

3.2 MaxScale REST API authentication approach

3.2.1 Current authentication method

The default local host address of MaxScale REST API is <http://127.0.0.1> and it listens on port 8989. In order to facilitate the development process easier, the address and port will be added to a global variable environment named VUE_APP_API in the “.env.development” file.

```
VUE_APP_API=http://127.0.0.1:8989
```

At the moment, MaxScale uses Basic Authentication for REST API to authenticate the user when its resources accessed through HTTP requests. The resources comprise of maxscale, services, servers, filters, monitors, sessions and users. The client sends the request to MaxScale with request headers contain the “Authorization” header. The value of this header consists of the word “Basic” followed after by a space and a base64 encoded string for username and password.

By adding an “auth” object in axios request config option, axios automatically adds a Basic Authentication header to the request header. The password and username will be also encoded using base64 encoding schemes by axios. The syntax is as followed:

```
axios.get(`${VUE_APP_API}/maxscale`, { auth: { username: 'admin', password: 'mariadb' } })
```

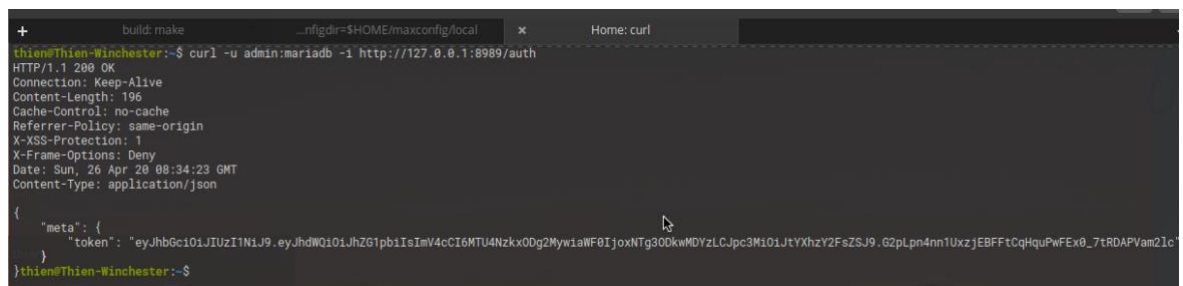
The request header added by axios will be then generated as follows: “Authorization: Basic YWRtaW46bWFyaWFkYg==”

3.2.2 JWT for SPA

With the current implementation of the REST API, SPA cannot keep the user logging in without storing credentials. Insecure application may store the credentials on the client side to achieve this. Though the connection is HTTPS encrypted, this approach is still an extremely bad practice and should be avoided because HTTPS does not compromise credentials leakage in all cases. Apart from that, the base64 encoded string can be decoded easily, hence the credentials is exposure unwittingly.

That leads to the need of developing an api endpoint “/auth” to authenticate and keep the user logged in by returning a token representing that user. As so, instead of exchanging credentials in every HTTP request, a token will be used. MaxScale implements stateless REST API which requires the session state to be shared between the client and MaxScale through HTTP protocol. The token plays a vital role as a state holding authentication data to be transmitted between MaxScale and MaxScale Admin GUI.

The current develop version of MaxScale is using JWT to create JSON-based access token which lasts for 8 hours and becomes invalid if MaxScale is restarted (MariaDB Corporation 2020). To get the token, a post request method with credentials enclosed in the body need to be sent to the “/auth” endpoint of the REST API. This “/auth” endpoint still uses the Basic Authentication schemes to authenticate the user, if the credentials are valid, the response body for the request contains the token (figure 21). After successfully login, for every future API request to MaxScale, the token needs to be sent along with the requests; otherwise MaxScale will not accept the request and return unauthorized http response code.



```
thien@Thien-Winchester:~$ curl -u admin:mariadb -1 http://127.0.0.1:8989/auth
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 196
Cache-Control: no-cache
Referrer-Policy: same-origin
X-XSS-Protection: 1
X-Frame-Options: Deny
Date: Sun, 26 Apr 20 08:34:23 GMT
Content-Type: application/json

{
  "meta": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXbzI1NiIsImV4cCI6MTU0NDg2Myw1aWF8Ij09eyJ0b2RkMDYzLCJpc3M101JtYXh2Y2FsZS9uG2pLpn4nn1UxzjEBFFtCqHquPwFEx0_7tRDAPVam21c"
  }
}
thien@Thien-Winchester:~$
```

Figure 21. JWT received from MaxScale "/auth" endpoint.

This token should be cryptographically signed and have expiration date so that it will not be modified by the client. It is tempting to store the token in the sessionStorage or

localStorage of the web storage due to the sake of simplicity. However, this is not the most secure approach for storing the token due to MITM, XSS and CSRF attacks. In fact, it is never a good idea to store any sensitive information in the web storage, whether it is in sessionStorage or localStorage, they are all vulnerable and accessible by JavaScript. A practical case is when the user may use their favorite browser extensions not knowing it contains malicious scripts. Attacker may steal the token and act on behalf of the user to perform unwanted actions. This type of attack is known as XSS, it bypasses the same-origin policy set by MaxScale REST API (OWASP Cheat Sheet Series 2020).

3.2.3 Storing the token

On the internet, there are so many articles, tutorials, discussions about storing the token. However, the use cases may vary ranging from the traditional web application to SPA. So far, one of the recommended approaches of storing the token in SPA is by using http-only cookie (Peter, L 2017).

Cookie with http-only flag prevents it from the access of JavaScript, as so it mitigates the XSS attack. Nevertheless, storing the token inside a http-only cookie without any configuration is still vulnerable to CSRF attacks and ineffective for UX in terms of SPA to some extent as follows.

Regarding the CSRF attack, though MaxScale recommend consuming REST API over https encrypted connection, this does not compromise CSRF prevention for storing a token in a http-only cookie. If user browses malicious websites, attacker can steal user's session and trick user to perform forgery requests as the cookie is shared among browser's tab (OWASP Foundation 2020).

Fortunately, a flag called 'SameSite' for cookie mitigates CSRF attack which have three option values: Strict, Lax and None. Although, this flag supports only well-known browsers with specific versions. Nevertheless, most modern browsers are migrating to force the use of SameSite cookie to defense CSRF attacks. For instance, since version 80, Chrome treats cookie as SameSite=Lax by default which prevents third-party context from accessing the cookie across sites (MDN web docs 2020).

As for the case of storing the token in http-only cookie, set SameSite=Strict is the most appropriate solution to mitigate CSRF attack in the GUI of MaxScale regardless of browser compatibility and sharing token with third-party context due to some reasons as follows:

- SameSite=strict flag supports modern browsers with versions older than MaxScale’s GUI does (browserl.ist 2020). Even so, the target browsers of the application still cover 90.77% of global browsers which is a remarkable figure (table 2).
- The token serves the application as an authentication layer to MaxScale REST API resources and it means not to be sent to any third-party context. If the application is served via <http://127.0.0.1:8989>, the token should only be sent to this address.

Table 2. The minimum versions support SameSite=Strict flag cookie compares to the browser versions MaxScale’s GUI support (MDN web docs 2020, browserl.ist 2020).

Browsers	SameSite=Strict		browserslist defaults	
	Mobile version	Desktop version	Mobile version	Desktop version
Chrome	51	51	78	78
Firefox	60	60	68	68
Edge		16		17
IE	NO	NO	11	11
Opera	41	39	46	63
Safari	12.2	12	12.2	12.1
Samsung Internet	5.0		9.2	
Android webview	51		76	

Another flag called “Secure” should be added along with the cookie which ensures the cookie will only be sent to the user if the transmission requested is encrypted using SSL or TLS. By adding this flag, in prevent the user’s session from being stolen by MITM attack.

When the token is set in the cookie with http-only flag, JavaScript will not have any access to the token. Though this guarantees the token integrity whenever the front-end send requests to MaxScale, SPA cannot verify the existence of the token in an http-only cookie which leads to the case that after logging in, SPA will not navigate to the dashboard. Successful http response code receiving after authenticating the user may tackle this

matter. However, SPA cannot rely fully on http response status code due to the characteristics of SPA.

For example, to restrict certain view based on user's role, SPA needs to use user's session data to manipulate this behavior. Sending HTTP request to MaxScale just to verify user's authentication is ineffective and considered to be bad practice. What can be done on client side should be done there. Though, MaxScale does not support user roles and access permissions at the moment, it may change since the administration GUI provides appealing UI than the CLI which is mainly used by administrator. The GUI means to be used by non-technical person; therefore, user role-based access control may be introduced.

Another scenario when SPA wants to handle user's session timeout in a better user-friendly approach which popping up a dialog to inform inactivity session and require re-login or just an animation of navigating to the login page. This is undoable if the token is stored with http-only flag since the application cannot access the token.

A solution to solve all the above-mentioned issues is to split the http-only cookie into two cookies which is the same approach as Peter L does (Peter, L 2017). A JWT token consists of three parts separated by dots after: header.payload.signature.

The first cookie name "token_body" holds the payload part and allows JavaScript to access it. This "token_body" contains information about the current authenticated user that can be used by SPA to achieve session timeout feature and page routing to dashboard page.

The other cookie name "token_sig" contains the signature part that MaxScale will use to verify user's authorization. The expiration time will be added only if the user chooses to use "Remember Me" feature, otherwise, when user closes the browser, this cookie will be invalid.

To wrap everything up, the cookies MaxScale would send to the GUI are listed as follows

```
Set-Cookie: token_sig =<signature>; SameSite=Strict; Secure; HttpOnly; Expires=<date>  
Set-Cookie: token_body = <payload>; SameSite=Strict; Secure; Expires=<date>
```

The new authentication mechanism flow when using two cookies approach is illustrated in figure 22 which describes the process for authenticating the user for the first time and when user is authenticated.

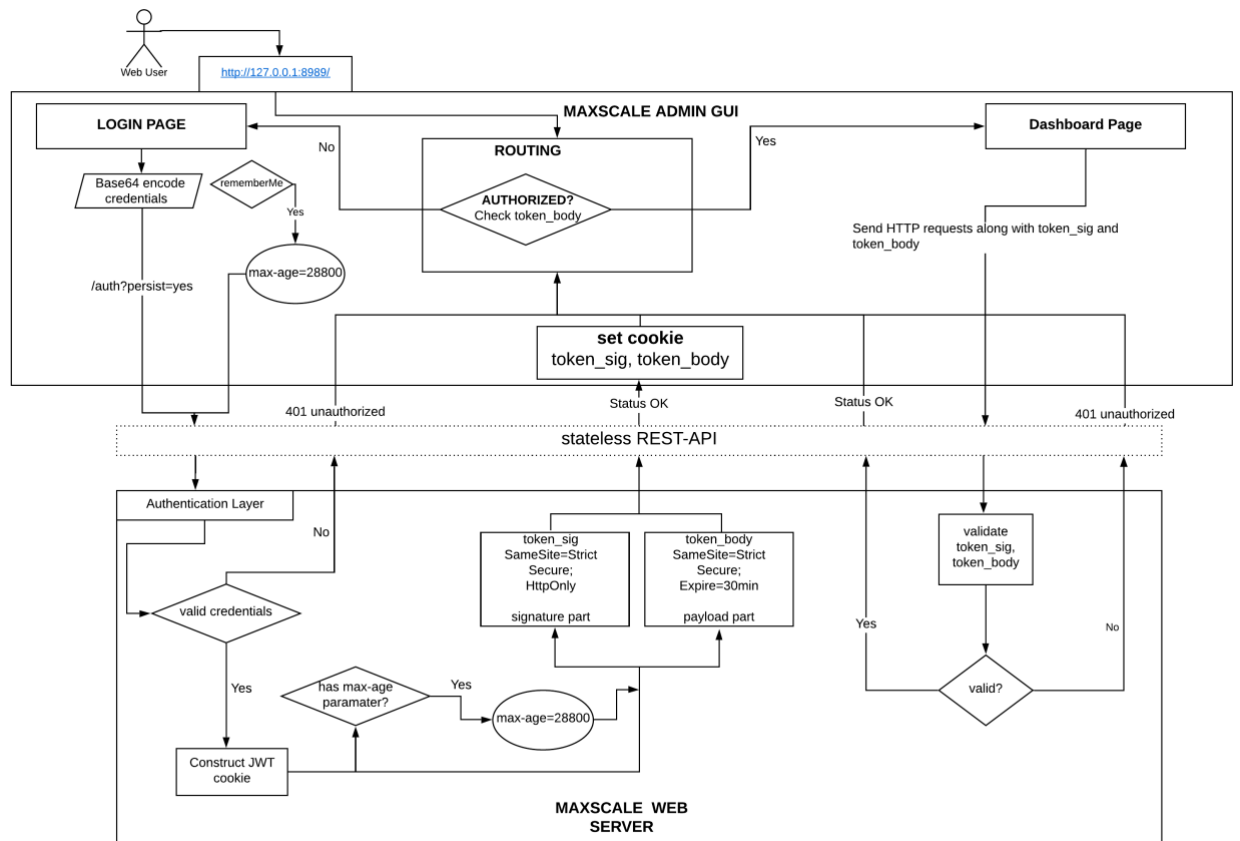


Figure 22. Authentication mechanism when storing token in two cookies.

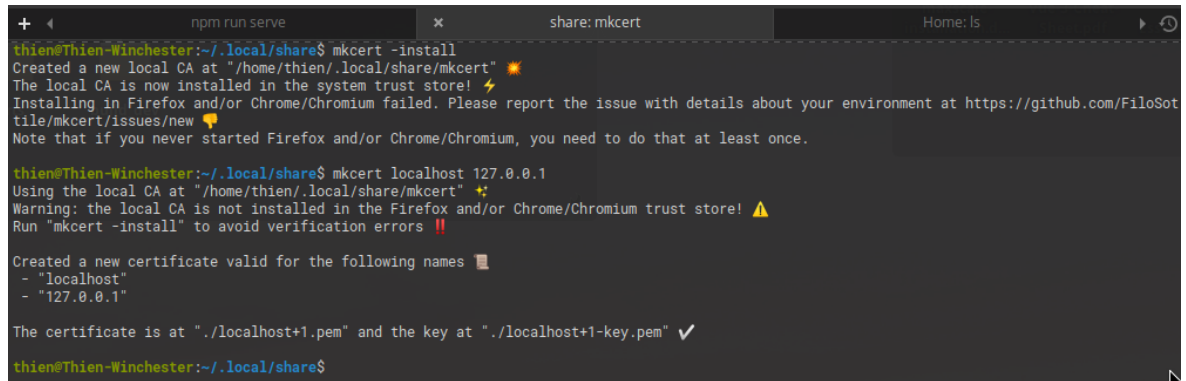
3.3 Implement the graphical user interface of authentication page

3.3.1 SSL encryption

As discussed in 3.2.3, the cookies will be sent only if the connection between MaxScale and the GUI application is encrypted when using the “Secure” cookie attribute. Although MaxScale will serve the GUI application from its origin which means that it only requires the configuration of TLS/SSL encryption in MaxScale; when developing the GUI application or hosting the application in localhost environment, the GUI and MaxScale are not in the same origin, the GUI is hosted at different address. Therefore, the following content will focus on setting up SSL encryption for both MaxScale and localhost environment.

For the sake of simplicity and speed, CA (certification authority) certificate which is a digital certificate issued by trusted CA will be created locally using open source “mkcert” tool created by Filippo, V. The browser uses this certificate to verify trusted CA to validate secure encryption. (Wikipedia 2020, Filippo, V. 2020)

The steps are quite simple as shown in figure 23, the “mkcert -install” simply create local trusted CA which will be then used to issue CA certificate for “localhost” and “127.0.0.1” addresses.



```
thien@Thien-Winchester:~/local/share$ mkcert -install
Created a new local CA at "/home/thien/.local/share/mkcert" ✨
The local CA is now installed in the system trust store! ⚡
Installing in Firefox and/or Chrome/Chromium failed. Please report the issue with details about your environment at https://github.com/FiloSottile/mkcert/issues/new 🙏
Note that if you never started Firefox and/or Chrome/Chromium, you need to do that at least once.

thien@Thien-Winchester:~/local/share$ mkcert localhost 127.0.0.1
Using the local CA at "/home/thien/.local/share/mkcert" ✨
Warning: the local CA is not installed in the Firefox and/or Chrome/Chromium trust store! ⚠️
Run "mkcert -install" to avoid verification errors ❗

Created a new certificate valid for the following names 📄
- "localhost"
- "127.0.0.1"

The certificate is at "./localhost+1.pem" and the key at "./localhost+1-key.pem" ✓

thien@Thien-Winchester:~/local/share$
```

Figure 23. Creating local CA certificate.

Regarding of TLS/SSL encryption for MaxScale, this can be done by configuring the configuration file “*.cnf” with the following parameters: admin_ssl_key and admin_ssl_cert holding value to the path of the CA certificate key (localhost+1-key.pem) and CA certificate (localhost+1.pem) files, respectively (MariaDB Corporation Ab 2020)

For the GUI application, enabling SSL can be done easily with similar steps which also requires specifying the path for both mentioned files. As mentioned in 3.1.2, figure 17, the “devServer” object which is used to configure the development server that host the GUI application in localhost environment; by adding a property named “https” to devServer object as below:

```
https: {
  key: fs.readFileSync('./.certs/localhost+1-key.pem'),
  cert: fs.readFileSync('./.certs/localhost+1.pem'),
}
```

However, specifying the path is not enough, we need to use “fs” API from nodejs to read and return the content of the file. Webpack starts the development server under the hood by passing devServer object to Node.js HTTPS module (Webpack 2020). Apart from that,

these certificate and key files are meant for using in local development which requires the local CA to be stored in developer's OS. Therefore, it is a good idea to store it in a directory called ".certs" at the root project directory and configure that folder to be ignore by GIT.

After successfully configuring SSL both MaxScale and the GUI are now only available to be accessed with https protocol. MaxScale REST API becomes: <https://127.0.0.1:8989> and the GUI is now served at <https://localhost:8000>. As so, the VUE_APP_API node.js environment variable as mentioned in [3.2.1](#) will be changed to use https instead of http protocol as follows:

```
VUE_APP_API=https://127.0.0.1:8989
```

3.3.2 Login page

The new JWT authentication approach still utilizes basic authentication in the first login request, therefore, user needs to provide credentials and send it to MaxScale via REST API at "/auth" endpoint as illustrated in figure 22. Besides, to remain consistency regarding user experience among MariaDB UI applications, the login page usually offers "remember me" feature. As so, the inputs collected from user consisting of username, password and remember me values which is illustrated in figure 24

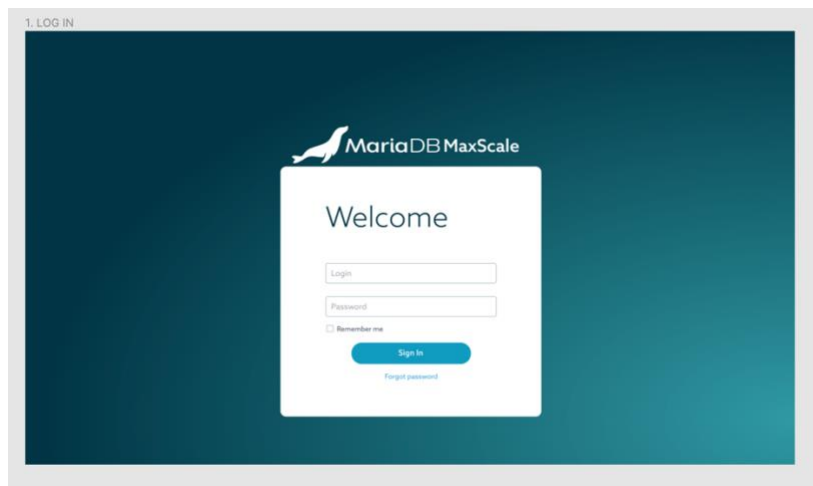


Figure 24. Login prototype page (MariaDB Corporation Ab 2020)

The following bullet points shown of the minimum requirements needed for validating user input and authenticating user:

- Input fields should not be empty, if it is empty, show related error message.

- If username or password are incorrect, shows a general error: “Incorrect password or username” and put two input fields into error state.
- Show server error message
- When “Remember me” checkbox is ticked, user’s session lasts for 8 hours. Otherwise, as long as user closes the browser, the session is expired.

With the traditional way using vanilla Javascript, in order to handle form submission, these input fields will be located in a html form tag `<form></form>`. This form has an “onsubmit” attribute that provides a callback function which is triggered when user clicks submit button. This callback function can be a function that validates form data and then attaches it in the body request to send to the web server. Regarding of collecting form data, there are several ways using vanilla Javascript to get values from the `<form>` tag such as using FormData API, adding event listener to the form or `querySelector()` method from the Document interface (MDN web docs 2020). Each approach has different scenarios usage case coming with its advantages as well as disadvantages.

Nonetheless, when using UI library or framework, handling form submission will be different depending on library or framework. In terms of a SPA framework, Vue.js handling form inputs using two-way data bindings approach which means when the input values in the form get changed by user’s interaction, the properties in the data model binding to the input value will be updated, vice versa (Vue.js 2020).

Since, the project was initialized by using Vue CLI, single file components was chosen by default which means files having named ending with “.vue” extension are reusable Vue instances. In other words, these files are called components which are located, organized in a structure of tree but still having access to the same options that root Vue Instance has.

Therefore, all components will have its own reactivity data, computed, watch, method as well as lifecycle hooks.

As so, to start the implementation of the login page, we first create a file named “Login.vue” in the `/src/views` directory which was created automatically when generating the project with Vue CLI using vue-router as SPA routing plugin. We can then later configure routing behavior in `/src/router` directory to achieve authenticated route and public routes. Below is a code snippet showing the login component’s data option which returns a reactivity data object for each component. As a rule of Vue.js, “data” has to be

always a function returning an object in order to prevent data changes in a specific component will not affect other components (Vue.js 2020).

```
data() {
  return {
    isValid: false,
    rememberMe: false,
    credential: {
      username: "",
      password: "",
    },
    errorMessage: "",
    showEmptyMessage: false,
    rules: {
      username: [val => !!val || this.$t('errors.usernameRequired')],
      password: [val => !!val || this.$t('errors.passwordRequired')],
    },
  }
},
```

Basically, the returned data object contains data binding we need to handle form submission. These properties in the object will be passed to the value of either v-bind or v-model directives in the HTML-based template, which can be called as reactivity data as mentioned in [2.1.1](#). The HTML-based template will be rendered as Virtual DOM performing necessary re-render if the values of those properties are changed. Directories v-bind and v-model are one-way data bindings and two-way data bindings, respectively. As explained before, in order to handle form inputs, we need to use two-way data bindings directive which is “v-model”. Apart from that, for one-way data bindings which is suitable for read-only data, we can use “v-bind” directive; For example: to display placeholder for input field, the syntax for it would be as so:

```
v-bind:placeholder="username"
```

As mentioned at the beginning of this section, the implementation of the login page needs to meet four mentioned requirements. To achieve the first bullet point requirement, we add required attribute for both input fields and specify array of validation methods for “rules” props of <v-text-field>, a Vuetify input component. By doing this way, validation method specified in “rules” props will receive the value from the associated input field to either return Boolean values or an error string message.

For example with rules.username method for username input field, this validation method receives a parameter value that I named it as “val”.


```
rules: {
  username: [val => !!val || this.$t('errors.usernameRequired')],
},
```

This “val” is a two-way data binding value, when user changes value of the input, it triggers this validation method which returns either Boolean value or a string. Therefore, if the Boolean value is false, it switches to return the error string as shown above.

To clarify this validation method, assuming the value is always empty, then we want to check the truthy of the value and it should be false so as to return the error string.

First, to put the string into Boolean evaluation mode, we add a not operator “!” before the value, so now empty string will be treated as a falsy value and the operator not “!” will convert it from false to true. But it goes against our will, the value is empty and is evaluated to be true, then the string error message will never be returned. That is why we need to add double not “!” operator, the second not operator simply inverts it to the original value which is false. Though the use of double not operator minifies the number line of code but makes it complex for those who does not understand the meaning of double not operator.

If we use simpler validation method, it would be written as follows:

```
rules: {
  username: [
    val => {
      if (val === "") {
        return this.$t('errors.usernameRequired')
      } else return true
    },
  ],
}
```

Regarding the second and the third bullet point requirements, we can combine those requirements into one goal which is to allow custom error message to be shown after performing asynchronous request to MaxScale web server.

Fortunately, Vuetify `<v-text-field>` provide props called “error-messages” which accepts either string or array. Under the hood, `<v-text-field>` component evaluates the value of this props to a Boolean value, if the value is provided with a non-empty string or array of non-empty string, the input field will be put into error state and rendered provided error message. However, if we pass empty string to this prop, it will not trigger error state as empty string is treated as a falsy value.

To achieve this, we bind the “error-message” props with errorMessage property in the data object so that we can control the content of the error message based on the response body whether it returns 401 (wrong credentials) or server error. However, when it returns 401, both input fields will render the same error message which is not our goal. If we omit to use “error-message” props for one of the two input fields, the omitted one will not be put into error state. To handle this issue, simply adding a property in the data object called “showEmptyMessage”. The value of this property is a boolean value that controls the display of empty message passed to “error-message” props, as follows:

```
v-bind:error-messages="showEmptyError ? ' ' : errorMessage"
```

If “showEmptyError” is true, we return a string with a space instead of the actual error message which we already shown in the other input. We return a space to bypass the empty string issue as mentioned above. At this stage, the error state is triggered, but when user tries to correct the error by re-entering the input field in term of wrong credential error, “error-message” props will not trigger form validation, hence the error state still visible in the user interface. A solution to tackle this problem is to detect the event when user updates input values. To watch on user input state, we add a v-on directive event for input as so: v-on:input=“onInput”, so whenever input is updated and the “showEmptyMessage” is evaluated to be true, we set the “errorMessage” to be an empty string and then clear “errorMessage” value and set “showEmptyMessage” to be true as shown below:

```
onInput() {  
  if (this.showEmptyError) {  
    this.showEmptyError = false  
    this.errorMessage = ""  
  }  
},
```

Finally, we can achieve the “remember me” feature by adding a new property called rememberMe to data object and then use v-model to have two-way data bindings to a Vuetify checkbox component as follows:

```
<v-checkbox v-model="rememberMe" class="small mt-2 mb-4" :label="$t('rememberMe')"  
color="primary" hide-details />
```

When user clicks sign in button, “handleSubmit” method as shown below will be triggered to handle form submission. We can then collect all user input in the data object including

the “rememberMe” property to either include the “max-age” parameters to “/auth” endpoint to get two cookies token as is already illustrated in figure 22.

```
async handleSubmit() {
  let self = this
  try {
    let url = '/auth?persist=yes'
    await self.axios.get(`${url}${self.rememberMe ? '&max-age=28800' : ''}`, {
      auth: self.credential,
    })
    await self.$router.push(self.$route.query.redirect || '/dashboard/servers')
  } catch (error) {
    this.showEmptyError = true
    this.errorMessage =
      error.response.status === 401 ? this.$t('errors.wrongCredentials') : error.response.statusText
  }
}
```

4 Discussion

This chapter discusses around the problem encountered while writing this thesis and presents known limitation and suggests for further improvement of researched solution in terms of REST API authentication methods.

4.1 Problem encountered

Frist of all, it is related to unfamiliar technology using for the product. By the time I was recruited to the company, which was on February, I was using React.js as a UI library to develop web applications. However, the company wants to remain technology consistent between UI projects which are using Vue.js as UI framework. Stepping into the product project as a beginner in Vue.js, it definitely has an effect on my thesis project timeline due to learning curve in terms of writing theoretical part.

Secondly, due to time limited, early stage of collecting user requirements, documenting on practical usage case of the product; the structure, contents of the product are all in prototype phase. Therefore, changes to the design and the content displaying in the UI happens frequently which makes difficult to implement UI reusable components. Apart from that, the look and feel, the design style of the application is following MariaDB

product design system which is a confidential information. Therefore, UI styles implementation cannot be included, which leads to narrow the scope of this thesis.

4.2 Limitation of researched authentication approach

With the two cookies authentication approach, the “Remember me” feature was implemented simply without taking all security aspects into account. Whenever “Remember me” feature is enabled, user’s session is lasted up to 8 hours even when the browser is closed, though it brings convenience to user to some extent, it does not compromise on leaked access token fully. If user forgets to logout the application, those who have access to the computer can steal user’s session.

In addition, the two cookies authentication approach claims to prevent the common security issues for web application in theory as penetration testing have not been carried out.

4.3 Further research

For resolving “Remember me” feature issue, session cookies renewal approach can be a potential solution though it needs further research. Basically, expire time of the token will be set to 30 minutes instead of 8 hours when user chooses “Remember me” feature. It will be automatically renewed to 30 minutes if authenticated user keeps sending requests. Otherwise, whenever user’s session is expired, the application prompts a login dialog to ask for credentials, then the token can be renewed.

Apart from that, there could have been a section to compare the difference, pros and cons of authentication methods for REST API to clarify the reason behind using JWT method.

4.3.1 Summary

Overall, though I did not achieve objectives outlined in the project plan due to problems encountered mentioned in [4.1](#), I was able to make adjustment to the structure of the thesis at the early stage. New objectives of this thesis including the setting up of development environment, UI project structure and improvement of MaxScale REST API in terms of user authentication in web application were obtained. Apart from that, knowledge regarding to REST API authentication, database proxy and Vue.js were accumulated significantly throughout the research.

References

Alibaba Cloud 2020. ApsaraDB for MariaDB TX. URL: <https://www.alibabacloud.com/products/apsaradb-for-rds-mariadb?spm=a2c5t.10695662.1389108.279.7b3d12d1dSVvBC>. Accessed 17 March 2020.

alligator.io 2017. Understanding Vue.js Lifecycle Hooks. URL: <https://alligator.io/vuejs/component-lifecycle/>. Accessed 23 March 2020.

Amazon Web Services 2020. Amazon RDS for MariaDB. URL: <https://aws.amazon.com/rds/mariadb/>. Accessed 17 March 2020.

BackBlaze 2018. What's the Diff: VMs vs Containers. URL: <https://www.backblaze.com/blog/vm-vs-containers/>. Accessed 08 March 2020.

bestofjs 2019. 2019 JavaScript Rising Stars. URL: <https://risingstars.js.org/2019/en/#section-framework>. Accessed 22 March 2020.

browserl.ist 2020. browserl.ist. URL: <https://browserl.ist/?q=>. Accessed 21 April 2020.

Docker 2020. Overview of Docker Compose. URL: <https://docs.docker.com/compose/>. Accessed 12 April 2020.

Docker 2020. What is a Container? URL: <https://www.docker.com/resources/what-container>. Accessed 17 March 2020.

Docker Hub 2020. mariadb Docker Official Images. URL: https://hub.docker.com/_/mariadb/. Accessed 17 March 2020.

Docker Hub 2020. MariaDB Server Docker. URL: <https://hub.docker.com/r/mariadb/server>. Accessed 17 March 2020.

Filippo, V. 2020. mkcert. URL: <https://github.com/FiloSottile/mkcert>. Accessed 6 May 2020.

Google Cloud Platform 2020. MariaDB. URL:

<https://console.cloud.google.com/marketplace/details/google/mariadb>. Accessed 17 March 2020.

JAVASCRIPT REPORT 2017. How Is React Different from Vue? URL:

<https://jsreport.io/how-is-react-different-from-vue/>. Accessed 22 March 2020.

MariaDB 2019. Learn how MariaDB MaxScale works with multiple MariaDB instances for HA. URL: https://www.youtube.com/watch?v=Uk0HMIVvN3I&ab_channel=MariaDB.

Accessed 07 March 2020.

MariaDB Corporation 2020. Building MariaDB MaxScale from Source Code. URL:

<https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/Getting-Started/Building-MaxScale-from-Source-Code.md>. Accessed 11 April 2020.

MariaDB Corporation 2020. REST API. URL: [https://github.com/mariadb-](https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/REST-API/API.md)

[corporation/MaxScale/blob/develop/Documentation/REST-API/API.md](https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/REST-API/API.md). Accessed 19 April 2020.

MariaDB Corporation 2020. Setting up MariaDB MaxScale. URL:

[https://github.com/mariadb-](https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/Tutorials/MaxScale-Tutorial.md)
[corporation/MaxScale/blob/develop/Documentation/Tutorials/MaxScale-Tutorial.md](https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/Tutorials/MaxScale-Tutorial.md).

Accessed 11 April 2020.

MariaDB Corporation Ab 2017. MariaDB MaxScale Setup with Binlog Server and SQL

Query Routing. URL: <https://mariadb.com/resources/blog/mariadb-maxscale-setup-with-binlog-server-and-sql-query-routing/>. Accessed 07 March 2020.

MariaDB Corporation Ab 2020. About MariaDB Software. URL:

<https://mariadb.com/kb/en/about-mariadb-software/>. Accessed 07 March 2020.

MariaDB Corporation Ab 2020. Enterprise Database Products. URL:

<https://mariadb.com/products/>. Accessed 07 March 2020.

MariaDB Corporation Ab 2020. Enterprise Database Products. URL:

<https://mariadb.com/products/>. Accessed: 23 February 2020.

MariaDB Corporation Ab 2020. MariaDB Enterprise. URL:
<https://mariadb.com/kb/en/mariadb-enterprise/>. Accessed 23 February 2020.

MariaDB Corporation Ab 2020. MariaDB MaxScale Configuration Guide. URL:
<https://mariadb.com/kb/en/mariadb-maxscale-24-mariadb-maxscale-configuration-guide/>.
Accessed 06 May 2020.

MariaDB Corporation Ab 2020. MariaDB MaxScale Installation Guide. URL:
<https://mariadb.com/kb/en/mariadb-maxscale-24-mariadb-maxscale-installation-guide/>.
Accessed 08 March 2020.

MariaDB Corporation Ab 2020. MariaDB MaxScale. URL:
<https://mariadb.com/kb/en/maxscale/>. Accessed 04 March 2020.

MariaDB Corporation Ab 2020. Running Multiple MariaDB Server Processes. URL:
<https://mariadb.com/kb/en/running-multiple-mariadb-server-processes/>. Accessed 17
March 2020.

MariaDB Foundation 2016. MariaDB Server is a true open source project. URL:
<https://mariadb.org/mariadb-true-open-source-project/>. Accessed 07 March 2020.

MariaDB Foundation 2020. MariaDB Server: The open source relational database. URL:
<https://mariadb.org/>. Accessed 04 March 2020.

MariaDB Foundation GitHub 2019. GNU General Public License v2.0. URL:
<https://github.com/MariaDB/server/blob/10.2/COPYING>. Accessed 07 March 2020.

MDN web docs 2020. Set-Cookie. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#Browser_compatibility. Accessed 21 April
2020.

MDN web docs 2020. Web APIs. URL: <https://developer.mozilla.org/en-US/docs/Web/API>.
Accessed 9 May 2020.

Oracle 1999. Distributed Processing. URL:
https://docs.oracle.com/cd/F49540_01/DOC/server.815/a67781/c29dstpr.htm. Accessed
07 March 2020.

OWASP Cheat Sheet Series 2020. Storage APIs. URL: https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#storage-apis. Accessed 20 April 2020.

OWASP Foundation 2020. Cross Site Request Forgery (CSRF). URL: <https://owasp.org/www-community/attacks/csrf>. Accessed 20 April 2020.

OWASP Foundation 2020. Cross Site Scripting (XSS). URL: <https://owasp.org/www-community/attacks/xss/>. Accessed 20 April 2020.

Peter, L 2017. Getting Token Authentication Right in a Stateless Single Page Application. URL: <https://medium.com/lightrail/getting-token-authentication-right-in-a-stateless-single-page-application-57d0c6474e3>. Accessed 20 April 2020.

React 2020. React.Component – React. URL: <https://reactjs.org/docs/react-component.html#shouldcomponentupdate>. Accessed 22 March 2020.

Reactjs.org 2020. Fragments. URL: <https://reactjs.org/docs/fragments.html>. Accessed 18 April 2020.

Reactjs.org 2020. React.Component. URL: <https://reactjs.org/docs/react-component.html>. Accessed 23 March 2020.

Science Direct 2020 Database Server. URL: <https://www.sciencedirect.com/topics/computer-science/database-server>. Accessed 07 March 2020.

Severalnines 2018. Choosing a Database Proxy for MySQL and MariaDB. URL: <https://severalnines.com/resources/whitepapers/choosing-database-proxy-mysql-and-mariadb>. Accessed 07 March 2020.

StackExchange UNIX&LINUX 2013. What are .deb and .rpm and how are they different from .msi? <https://unix.stackexchange.com/questions/103531/what-are-deb-and-rpm-and-how-are-they-different-from-msi>. Accessed 08 March 2020.

Thien, L. 2020. 'Is it feasible to install MaxScale in MacOS operating system?'. Haaga-Helia University of Applied Sciences. Unpublished report assignment.

TWILIO INC 2017. Working with Environment Variables in Node.js. URL: <https://www.twilio.com/blog/2017/08/working-with-environment-variables-in-node-js.html>. Accessed 16 April 2020.

Vue CLI 2020. Browser Compatibility. URL: <https://cli.vuejs.org/guide/browser-compatibility.html>. Accessed 21 April 2020.

Vue NYC 2017. VueNYC - Vue.js: the Progressive Framework - Evan You - YouTube. URL: https://www.youtube.com/watch?v=p2P3z7p_zTI&ab_channel=VueNYC. Accessed 22 March 2020.

Vue.js 2020. API. URL: <https://vuejs.org/v2/api/#updated>. Accessed 25 March 2020.

Vue.js 2020. Components Basics. URL: <https://vuejs.org/v2/guide/components.html>. Accessed 9 May 2020.

Vue.js 2020. Introduction — Vue.js. URL: <https://vuejs.org/v2/guide/>. Accessed 22 March 2020.

Vue.js 2020. Form Input Bindings. URL: <https://vuejs.org/v2/guide/forms.html>. Accessed 9 May 2020.

Vue.js 2020. Lifecycle Diagram. URL: <https://vuejs.org/v2/guide/instance.html#Lifecycle-Diagram>. Accessed 23 March 2020.

Vue.js 2020. Modes and Environment Variables. URL: <https://cli.vuejs.org/guide/mode-and-env.html#modes>. Accessed 16 April 2020.

Vue.js 2020. Reactivity in Depth. URL: <https://vuejs.org/v2/guide/reactivity.html>. Accessed 22 March 2020.

Vue.js 2020. Using Axios to Consume APIs. URL: <https://vuejs.org/v2/cookbook/using-axios-to-consume-apis.html>. Accessed 18 April 2020.

Vue.js GitHub 2017. Allow more than 1 root element for Template. URL: <https://github.com/vuejs/vue/issues/7088>. Accessed 18 April 2020.

Vue.js GitHub. 2.0 Changes. URL: <https://github.com/vuejs/vue/issues/2873>. Accessed 25 March 2020

Webpack 2020. DevServer. URL: <https://webpack.js.org/configuration/dev-server/#devserverhttps>. Accessed 06 May 2020.

Widenius, M. 2008. Sun buys MySQL AB. URL: <http://monty-says.blogspot.com/2008/01/sun-buys-mysql-ab.html>. Accessed: 23 February 2020.

Widenius, M. 2010. Welcoming SkySQL, a new home for MySQL talents. URL: <http://monty-says.blogspot.com/2010/07/welcoming-skysql-new-home-for-mysql.html>. Accessed: 23 February 2020.

Widenius, M. 2014. Why SkySQL becoming MariaDB Corporation will be good for the MariaDB Foundation. URL: <http://monty-says.blogspot.com/2014/10/why-skysql-becoming-mariadb-corporation.html>. Accessed: 23 February 2020

Wikipedia 2020. Ajax (programming). URL: [https://en.wikipedia.org/wiki/Ajax_\(programming\)](https://en.wikipedia.org/wiki/Ajax_(programming)). Accessed 18 April 2020.

Wikipedia 2020. Certificate authority. URL: https://en.wikipedia.org/wiki/Certificate_authority. Accessed 6 May 2020.

Table of figures

Figure 1. Diagram of reactivity system in Vue.js (Vue.js 2020)	7
Figure 2. Data object in vue instance (Vue.js 2020)	8
Figure 3. Object and Array changes detection	8
Figure 4. React.js Lifecycle diagram (Reactjs.org 2020)	10
Figure 5. Vue Lifecycle diagram (Vue.js 2020)	11
Figure 6. MariaDB Platform X4 (MariaDB Corporation Ab 2020)	12
Figure 7. The Client/Server Architecture and Distributed Processing (Oracle 1999)	13
Figure 8. MariaDB MaxScale (MariaDB Corporation Ab 2017)	15
Figure 9. VM architecture and Container architecture (Docker 2020)	16
Figure 10. MaxScale building from source code build steps (MariaDB Corporation Ab 2020)	18
Figure 11. MaxScale build options for minimum build version.....	19
Figure 12. Running MaxScale.	20
Figure 13. Basic MaxScale configuration file	20
Figure 14. docker-compose.yml file.....	21
Figure 15. Persisting and sharing data between primary server and slave server	21
Figure 16. Output of the "maxctrl list servers" command.	22
Figure 17. vue.config.js.....	23
Figure 18. Resolve path to use module path.....	24
Figure 19. Visual Studio Code configuration	26
Figure 20. Code snippet of vue-axios and the usage of it within Vue.js.	27
Figure 21. JWT received from MaxScale "/auth" endpoint.	29
Figure 22. Authentication mechanism when storing token in two cookies.	33
Figure 23. Creating local CA certificate.	34
Figure 24. Login prototype page (MariaDB Corporation Ab 2020)	35