

Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria
Master of Science in Telecommunication Engineering



License monitoring tool design for Toshiba Global Commerce Solution Italy

Supervisor:

Prof. Alessandro Enrico Cesare Redondi

Candidate:

Ehsan Nayernia

Matricola: 926902

ACADEMIC YEAR 2020-2021

ACKNOWLEDGEMENT

Nobody has been more important to me in pursuing this project than my lovely wife, who accompanied me during my master's education.

I am grateful to my kind parents, who support me a lot.

This work would not have been possible without the Toshiba global commerce solution Italy, especially Stefano Ales, and Ermelinda Rapolli, my supervisors, who let me participate in this project.

In the end, I should thank my supervisor at Politecnico di Milano, Prof. Alessandro Enrico Cesare Redondi, for his help and feedback.

These have proven to be instrumental in completing my master's degree, thereby achieving my goals.

ABSTRACT

All released paid software programs in the world need a license monitoring tool to play a part of the security role for the software owner. As an IT administrator or software owner, you need a tool to keep track of software usage and license compliance. Depending on the main production, the license monitoring tool design could be different. Many technology sets can be used during the development of the backend and frontend of license monitoring tools considering complexity, criticality, and the budget of the product owner. Also, since the product can be categorized into two main online and offline categories, the license monitoring tool could follow different design processes.

In this specific case, Toshiba Global Commerce Solution Italy has released a new version of its product, "VisualStore." Compared to the previews versions, they are trying to immigrate more to the network area, which provides more flexibility to the customers, which could be any kind of hyper stores. The main feature of the new version is that all devices available in the customer side can work under the territory of a central enterprise that can send all promotions, configurations, and services to its sub-devices. Obviously, all installed productions on both enterprise and all sub-devices must have a paid license.

In this situation, the solution I developed was an online web application that can do license monitoring from Toshiba side, which is connected to the installed software on the enterprise. The enterprise must report the license situation of itself and all other connected sub-devices to the license monitoring tool application. This online solution allows a continuous checking of licenses and stops any anomalies on the customer side like cloning.

ITALIAN ABSTRACT

Tutti i programmi software a pagamento rilasciati nel mondo necessitano di uno strumento di monitoraggio delle licenze per svolgere un ruolo di sicurezza per il proprietario del software. In qualità di amministratore IT o proprietario di software, hai bisogno di uno strumento per tenere traccia dell'utilizzo del software e della conformità delle licenze. A seconda della produzione principale, il design dello strumento di monitoraggio delle licenze potrebbe essere diverso. Molti set tecnologici possono essere utilizzati durante lo sviluppo del backend e del frontend degli strumenti di monitoraggio delle licenze considerando la complessità, la criticità e il budget del proprietario del prodotto. Inoltre, poiché il prodotto può essere classificato in due principali categorie online e offline, lo strumento di monitoraggio delle licenze potrebbe seguire diversi processi di progettazione.

In questo caso specifico, Toshiba Global Commerce Solution Italy ha rilasciato una nuova versione del suo prodotto, "VisualStore". Rispetto alle versioni in anteprima, stanno cercando di immigrare di più nell'area di rete, che offre maggiore flessibilità ai clienti, che potrebbero essere qualsiasi tipo di ipernegozio. La caratteristica principale della nuova versione è che tutti i dispositivi disponibili sul lato cliente possono funzionare nel territorio di un'impresa centrale che può inviare tutte le promozioni, le configurazioni e i servizi ai suoi dispositivi secondari. Ovviamente, tutte le produzioni installate sia su enterprise che su tutti i sub-dispositivi devono avere una licenza a pagamento.

In questa situazione, la soluzione che ho sviluppato era un'applicazione web online in grado di monitorare le licenze da parte di Toshiba, che è collegata al software installato nell'azienda. L'azienda deve segnalare la situazione della licenza propria e di tutti gli altri dispositivi secondari collegati all'applicazione dello strumento di monitoraggio delle licenze. Questa soluzione online permette un controllo continuo delle licenze e blocca eventuali anomalie lato cliente come la clonazione.

Contents

1	INTRODUCTION.....	1
1.1	THESIS AIM	3
1.2	OBJECTIVES	3
1.3	THESIS OUTLINE	3
2	THEORY PART	5
2.1	REST API	5
3	PRIMARY IMPLEMENTATIONS	12
3.1	DATABASE.....	12
3.2	USER INTERFACE.....	14
3.3	SECURITY	15
3.3.1	User registration flow.....	15
3.3.2	Assigning roles to each user	16
3.3.3	Assigning password and password encryption	16
3.3.4	Login overlay for VSCLS	17
3.3.5	Forgot and Reset password flow.....	18
3.3.6	Keep tracking of Logged-in user as Current user	20
3.4	SMTP IMPLEMENTATIONS	21
3.5	REPORTS IN PDF AND CSV.....	22
4	SECONDARY IMPLEMENTATIONS.....	24
4.1	IMPLEMENTING COMPANY AND ENTERPRISE TABLE.....	24
4.2	FIRST INSTALLATION OF VS ON ENTERPRISE AND ACTIVATION FLOW.....	28
4.3	WEEKLY REPORT.....	30
5	PERFORMANCE EVALUATION	33
6	CONCLUSION AND FUTURE WORKS	36
	REFERENCES.....	39

TABELS

<i>Table 1. Request Methods [2]</i>	9
---	---

FIGURES

Figure 1. Connection of VSCLS and Installed VS	1
Figure 2. General scheme of REST API [1]	6
Figure 3, Example of URI, Method and transited Information used in a REST API in VSCLS	8
Figure 4. Husband (Left) and Wife (Right) Tables with One-to-One Relation	13
Figure 5. One to many logical relation	13
Figure 6. Example of annotations used for routing in UI	14
Figure 7. Home page of application	15
Figure 8. User registration	15
Figure 9. Hiding Users Tab for a user with role "USER"	16
Figure 10. Annotation used to secure a page for "Admin" role	16
Figure 11. Binding Rules for User registration	17
Figure 12. Login overlay	18
Figure 13. Steps of forgot and reset password flow	20
Figure 14. Saving User id of the creator in Company table and showing with the name in the visual grid.	21
Figure 15. Example of developing a Many-to-one relation between Company table and User table	21
Figure 16. Sample Email sent via application and SMTP connection with Gmail account.	22
Figure 17. How to start Java mail sender utilities	22
Figure 18. PDF generation from Company Table	23
Figure 19. example of PDF generation from Company Table by searching companies by their country "Italy"	23
Figure 20. How to make a query on a table	23
Figure 21. Columns of Company table	24
Figure 22. Sample Code for Listing cities with Post Method	25
Figure 23. User Interface of Company Creation flow.	25
Figure 24. Columns of Enterprise table	26
Figure 25. Enterprise detail including activation code in PDF	27
Figure 26. UI for uploading a CSV containing the Companies information	28
Figure 27. First Installation and activation flow of enterprise	29
Figure 28. JSON message from enterprise to VSCLS for activation flow and the response of 200OK	30
Figure 29. Weekly report flow	32
Figure 30. 200 Ok Response to an existing enterprise with valid activation code.	34
Figure 31. 208 already Reported response to an already activated enterprise.	34
Figure 32. 404 Not found response to an activation code which is not existed in database	34
Figure 33. Visual warning against anomalies during three consecutive weeks	35
Figure 34. example of JWT method [6]	38

CHAPTER ONE

1 Introduction

VisualStore Centralized License monitoring system is an application based on Internet network for monitoring the licenses and their validity for software named VisualStore (“VS”) developed by Toshiba Global Commerce Solution, which is a POS platform characterized by a single central core, customizable for the various segments of the retail sector, including food, consumer electronics, and media, department stores, specialized stores, multi-store chains, and store-in-store operations. It offers centralized management and control that allow retailers to create promotions and manage processes, articles, and services from a single point. Furthermore, it is possible to configure and manage remotely, workstations POS, Smart Scales, Self-Scanning, Self-Checkout and Self-Payment, design and graphic user interface (GUI) updates included. VisualStore will be installed on Companies enterprise (single central core) and its sub-devices, as mentioned workstations above. While the devices are connected to each other in the company side network, they will be controlled from an enterprise system also installed on the company side. The enterprise must also be connected to the VisualStore Centralized License monitoring system (“VSCLS”) to constantly check for its license and its validity and for all sub-devices working under enterprise territory.

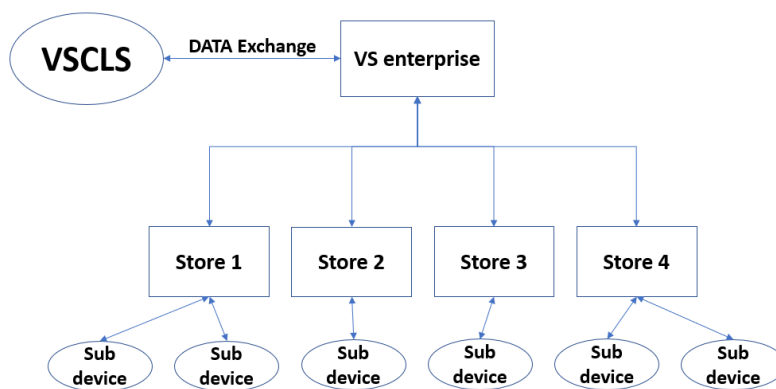


Figure 1. Connection of VSCLS and Installed VS

From a technology point of view, VSCLS is a web application. It is a Maven project for Java using Spring Framework and Vaadin Framework in its design. Combining these technologies gives us the ability to design a web application and develop backend and frontend simultaneously.

VSCLS primary duties are:

- Track how many active devices a customer has in their stores.
- How many licenses a customer still has, compared to the total number purchased on Oracle.
- Know immediately, via visual warning or notification, if a customer uses more devices than they paid for.
- Being able to perform actions in the face of anomalies found, automatic or manual action has to be decided.

VSCLS needs like every other web application, a body and instruction to start running on the web and makes its connections to all other application cooperating with. Designing standard Layout and UI, implementing databases, security of the application, REST connections, and SMTP are essential considerations for VSCLS implementation.

Representational state transfer (REST) is a software architectural style that uses a subset of HTTP. It is used to create an application based on web services and network. A Web service that follows these guidelines is called RESTful.

The Simple Mail Transfer Protocol (SMTP) stands for internet standard communication protocol for electronic mail transmission. Web applications like VSCLS and mail servers and any other message transfer operators use SMTP to send and receive mail messages.

So, in the rest of this article, I want to divide the report into two main parts:

1. Primary Implementations: fundamental developments for application as mentioned above for general aspects.
2. Secondary Implementations: It is related to the main duties of VSCLS and the flow of protocols and connections between installed VS on enterprise and VSCLS.

1.1 Thesis Aim

- develop backend and frontend of a web application whose main duty is a license monitoring tool.
- Analyse what can be developed as the further steps

1.2 Objectives

With the decision of Toshiba Global Commerce Solution to immigrate from VisualStore version 5 to version 6, it is necessary to design and implement a new license monitoring tool as described above. The main difference between versions 5 and 6 is the new definition of an enterprise in the system to play the role of the central core. Since the system is working based on the network, an online license monitoring could (web application) be a reasonable solution.

1.3 Thesis Outline

This thesis is organized in the following way:

Chapter 1 (Introduction) briefly introduces us to the concept of software released by Toshiba Global Commerce Solution, VisualStore, and the need for new license monitoring tool as VSCLS.

Chapter 2 (Theory part) briefly introduces the backbone theories that will be used during the development of VSCLS.

Chapter 3 (Primary implementation) fundamental developments for application as mentioned above for general aspects.

Chapter 4 (Secondary implementation) Related to main duties of VSCLS and the flow of protocols and connections between installed VS on enterprise and VSCLS.

Chapter 5 (Performance evaluation) Check if critical aspects of development are working properly.

Chapter 6 (Conclusion and future works) briefly doing an overview of what has been developed during this project, and as a proposal, two new features have been suggested for future jobs:

1. Rules against unauthorized enterprises
2. Implementing JWT method

CHAPTER TWO

2 Theory part

2.1 REST API

Representational state transfer (REST) is a kind of software architectural model which has been used for a subset of HTTP. It is used to create interactive applications that use Web services. A Web service that obeys these rules is called a RESTful application. For example, a Web service must provide a kind of content in a textual format, and they must be readable and can be modified with a stateless protocol and a predefined set of operations which means REST. This approach offers cooperation between the computer systems on the Internet (Server) that provide these services to the endpoints (Client). REST could be a good choice for other ways, such as SOAP, to access a Web service. Unlike SOAP, which is also based on web services, there is no "official" standard for RESTful web APIs. that is why we call REST an architectural style but SOAP a Protocol. REST is not a standard protocol but a RESTful application, and its implementations use standards, such as HTTP, URI, JSON, and XML.

Why REST? REST includes all the principles of the web, including its architecture, benefits, and everything else. REST has quickly become the most famous standard for developing web services on the web because they're easy to make and easy to consume. Let's quickly overview the features and benefits of using REST over the web's core protocol, which is HTTP. Appropriate actions (GET, POST, PUT, DELETE, ...)

- Caching
- Redirection and forwarding
- Security (encryption and authentication)

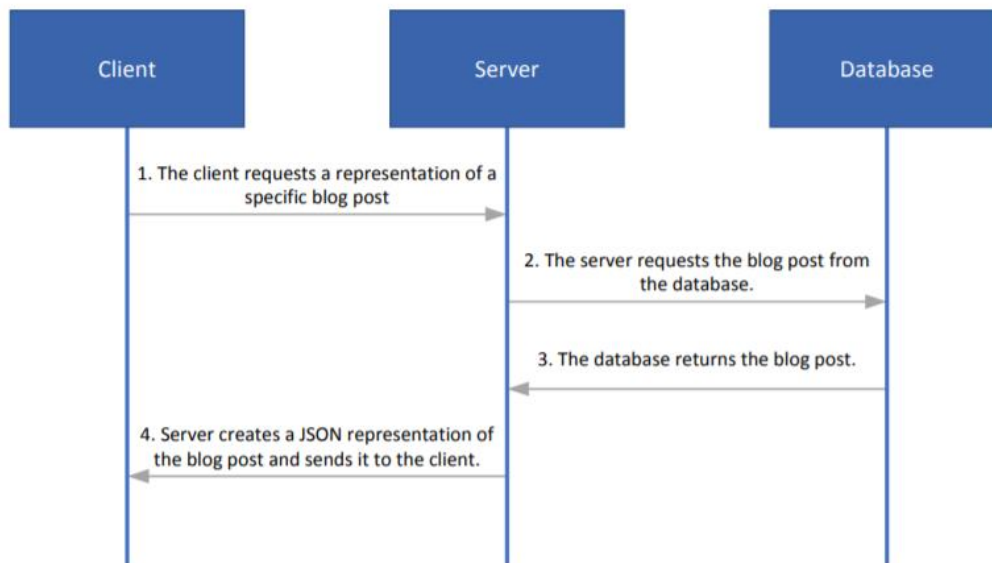


Figure 2. General scheme of REST API [1]

We are designing The REST architectural style for network-based applications, especially for a client-server application. However, even more, we use it for Internet-scale usage, so the connection between the user (client) and the server must be as light as possible to let us make large-scale applications based on the internet. It is done by creating a layer of abstraction on the server that encapsulates entities (e.g., files) on the server and hiding the underlying implementation details (file server, database, etc.). Nevertheless, the definition is even more general than that: any information or file that can have a name can be considered as a resource: an image, a database query, a temporal service like the status of weather in a city, or even a set of other resources. This approach allows excellent cooperation between clients and servers in a substantial Internet-scale environment.

The client can request a resource using a URI, and the server responds with a representation of the resource. A resource representation is another important concept in REST; to ensure the broadest possible number of clients can use responses, the server sends a representation of the resource in hypertext format. Thus, a resource is manipulated through hypertext, representations transferred in messages between the clients and servers.

The strong connection of client and server with the text-based transfer of information using a uniform addressing protocol is the initial steps for meeting the requirements of the web like robustness scalability, independent deployment of components, large data transfer, and a low-entry barrier for content readers.

To define an HTTP-based RESTful API, we need three main parts:

- A base URI like <http://api.example.com> (we will see in developing that we have a lot of usage of URIs for downloading and uploading files through REST. Example: <http://localhost:8080/users/export/pdf> will be explained in Reports in PDF and CSV)
- Standard HTTP methods
- Information, data, or any media type (in VSCLS, it is usually Json message format or PDF or CSV files) that wanted to be transited between server and client as the representation of the state. It is also usually called a payload.

In the figure below, which is a part of development for VSCLS (will be explained in the future), you can see the main three elements of a REST API explained above.

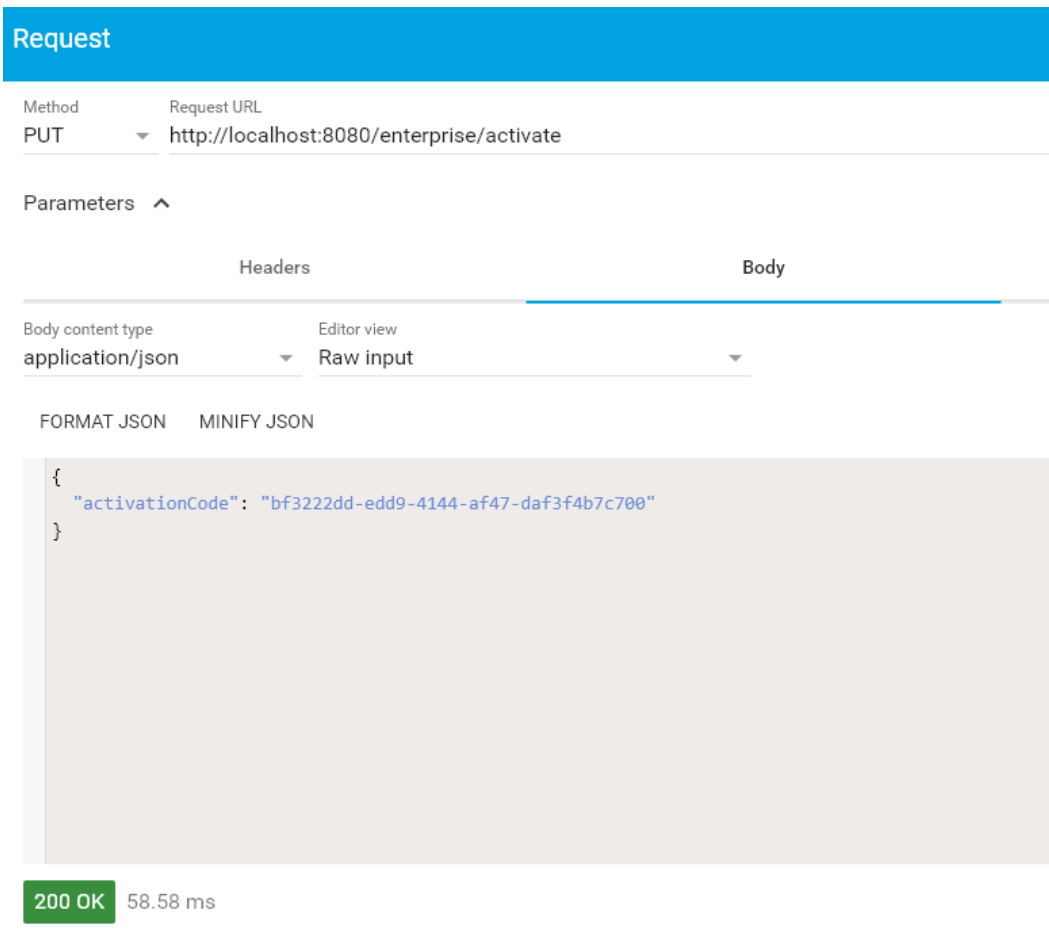


Figure 3, Example of URI, Method and transited Information used in a REST API in VSCLS

HTTP defines some method that shows the action that wants to be performed on the identified resource. This resource can represent preexisting or even dynamically produced data depending on the needs of client-server connections. Clients can use any of these methods, and the server can be configured in a way to respond in any of these methods or even a combination of them. In this article, four main methods that could be highly useful during the development of VSCLS are explained in the table below.

Table 1. Request Methods [2]

HTTP Method	CRUD equivalent	Safe	Description
GET	Read	Yes	Get a representation of the target resource's state.
POST	Create	No	Let the target resource process the representation enclosed in the request.
PUT	Update	No	Set the target resource's state to the state defined by the representation enclosed in the request.
DELETE	Delete	No	Delete the target resource state.

Those methods which cannot have any effect on the server-side can be called safe. In other words, a safe method is only readable.

Cacheable is an ability of a method if responses to the requested method could be storable for any use in other cases. In the general topic of all listed methods, GET, HEAD, and POST are cacheable. On the other hand, the methods PUT, DELETE, CONNECT, OPTIONS, TRACE, and PATCH are not cacheable.

Get method requests to transfer the information or data or selected representation for a target resource. Usually, when people are talking about retrieving data with HTTP, they are referring to Get request. Like downloading pdf or asking about the activation status of client from a server or listing all the countries and their cities from another RESTful application inside VSCLS, such a way users do not need to type the country and city in dedicated form, and it will be selectable. These are features that have been used in developing VSCLS. The payload of the Get method is meaningless according to the definition of the Get method.

The POST request method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it. [3] Against the Get method, its use case is usually when you want to upload data or submit a complete form. The payload of Post Method could be the set of information or data which is desired to be transferred.

The PUT method requests that the state of the target resource be created or replaced with the state defined by the representation enclosed in the request message payload. [3] The main difference between PUT and POST is that using the same PUT request several times makes the same result consistently. However, calling a POST request repeatedly creates the same resource multiple times.

In conclusion, I want to list some constraints that could be considered in the REST architecture and briefly explain and overview them.

1. Client-Server: the architecture is made of two sides which are served as a resource provider and client as a consumer
2. Stateless: the connection between client and server is stateless. This means that all information about the client's session is kept on the client, and the server knows nothing of it, so there should be no browser cookies, session variables, or other stateful features. Since the REST architecture is stateless, each request must contain all the necessary information that needs to be transferred among client and server
3. Cache: As we briefly explained above, each request must be categorized into two main categories of cacheable or noncacheable.
4. Uniform interface: What separates REST from other architectural styles is the Uniform Interface enforced by the fourth constraint. We don't usually think about it, but, amazingly, you can use the same Internet browser to read the news and to do your online banking, despite these being fundamentally different applications. You don't even need browser plug-ins to do any of this. [1]

Also, when we are talking about a uniform interface, it must also be considered that each resource must be identified with a unique URI on the server-side as we will see in the implementations part that each resource like downloading has its own URI.

Another point is the uniform body format of REST which is usually JSON, and this feature has also made a unicity for the interface of REST.

Using the standard methods (GET, PUT, etc.) lets REST have a uniform interface in its communications.

5. Layered system: The client only communicates with the immediate layer. It means that the client does not know anything about behind layers or even intermediate layers like a proxy between server and client. It also means that we can add security as a layer on top of the web services and then clearly separate business logic from security logic. [1] as we will see in the implementation part, we have provided security over our RESTful application. Each method must have this permission to have access to the server of VSCLS.

CHAPTER THREE

3 Primary Implementations

3.1 Database

Implementing the

the database is necessary for most web applications. In VSCLS, we use PostgreSQL, also known as Postgres, a free and open-source relational database management system emphasizing extensibility and SQL compliance. The maven project is well connected to the Postgres database via some configurations in maven dependencies, and the Java classes marked with @Entity will be created functional in the Postgres Environment. Here is the list of Entities Developed in the VSCLS, and all of them will be explained in their situations.

- ✓ Users
- ✓ Company
- ✓ Enterprise

In the design of each table in the database, it is considered to have an Id column as a Primary key. A primary key also called a primary keyword, is a key in a relational database that is unique for each record. It is a unique identifier, like a driver's license number, telephone number (including area code), or tax code in Italy (codice fiscale). A relational database must always have one and only one primary key. Setting an id for each row of the table is a key to proceed with relational tables.

In this application, a table could be the parent table while the other one could be related to Child, as the relation between company and Enterprise tablets. This is a one-to-one logic relation developed by annotations in java code @OneToOne. This means that each Row of the Company Table Could be related to only and only one row of the enterprise. It makes

sense that while we define a Company First, we can define an Enterprise dedicated to that company.

Here is an example of One-to-One relation for clarifying. Wife and Husband Could be an example of One-to-One relation. In the figure below, the relation is obvious.

id	name
1	David
2	Peter
3	Phillip
NULL	NULL

id	name	husband_id
1	Lisa	1
2	Marv	2
3	Lauren	3
NULL	NULL	NULL

Figure 4. Husband (Left) and Wife (Right) Tables with One-to-One Relation

The other good logical relation which could be helpful in future implementation is “@OneTOMany.” Here against the previews logical relation (@OneToOne), the one-to-many relationship is a type of cardinality that refers to the relationship between two entities A and B in which an element of A may be linked to many elements of B, but a member of B is linked to only one element of A. For instance, think of A as books and B as pages.

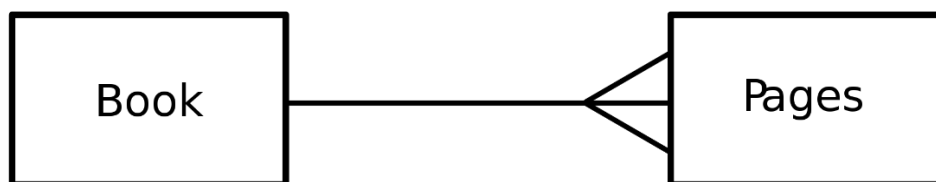


Figure 5. One to many logical relation

3.2 User interface

All the components on the user interface of the web application come from the Vaadin library. Thanks to Vaadin components, we can have good and reasonable relation between the backend and User Interface. With surfing on the application, it can be seen that all the components like Grids, Input fields, links, and buttons and generally the Layout which is based on `MainView.class` can give us a good experience of having a connection between user and backend. One of the most important aspects of this application is good Routing among different Tabs and pages. Each page has a sub address from the main address of `http://localhost:8080`. Annotating the Homepage class by `@RouteAlias` is the start point of the application after the login page and security. Different tabs and events in the application are addressed correctly, and navigation between them with Route address has been developed.

Here are some examples:

- ✓ <http://localhost:8080/HomePage>
- ✓ <http://localhost:8080/users>
- ✓ <http://localhost:8080/company>
- ✓ <http://localhost:8080/enterprise>

```
25 @Route(value = "HomePage", layout = MainView.class)
26 @PageTitle(Constants.TITLE_HOME_PAGE)
27 @CssImport("./styles/views/homePage.css")
28 @RouteAlias(value = "", layout = MainView.class)
29 @PreserveOnRefresh
30 public class HomePageView extends VerticalLayout {
```

Figure 6. Example of annotations used for routing in UI

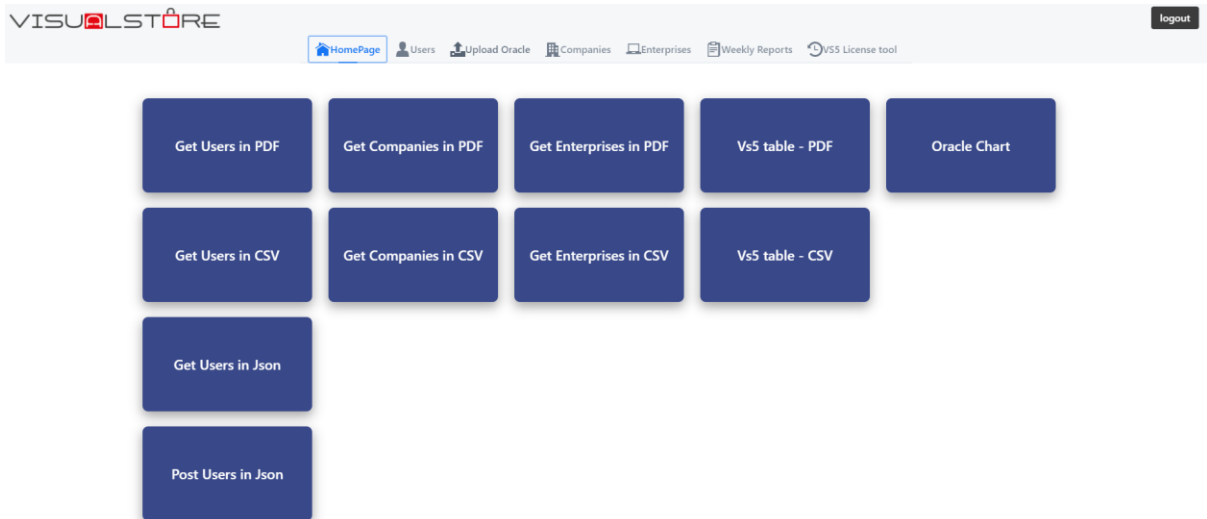


Figure 7. Home page of application

3.3 Security

To have a Secured app, step by step, these functionalities have been provided.

3.3.1 User registration flow

Based on the needs of VSCLS creating new users is only possible by a primary admin user. In our case, users are all Toshiba's internal employees. So, this application is not following any Sign-up flow, which is usually done by users. User registration flow could be done through the Users Tab shown below:

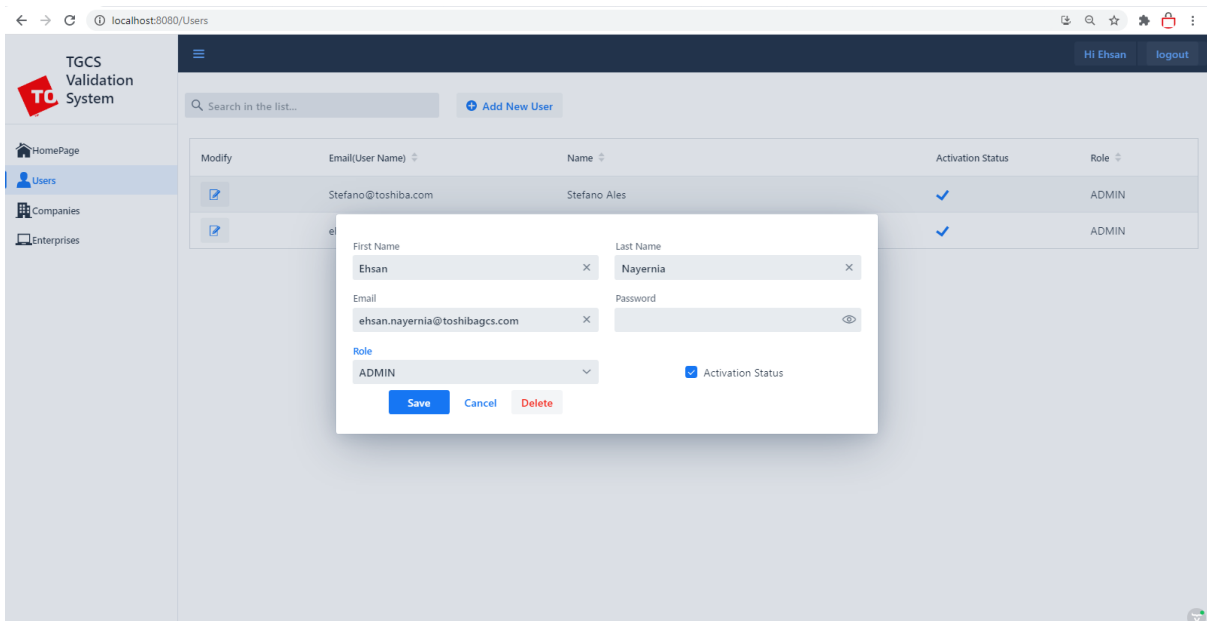


Figure 8. User registration

3.3.2 Assigning roles to each user

I have decided to add a useful feature of different user levels for this application during the user registration flow. Two roles have been provided “ADMIN” and “USER.” This will allow us to forbid or even hide some functionalities which can be done only by ADMIN role for the ordinary users. User registration is one of those functionalities that could be done only by only ADMIN role. Hiding the Users tab for a USER role can be seen in figure 6 by comparing with figure 5. This is a concept functionality that could be enforced wherever it is necessary for the future. Again, thanks to powerful annotations on top of UsersView.java and some other developments in SecurityUtils.java, this feature is working very well.

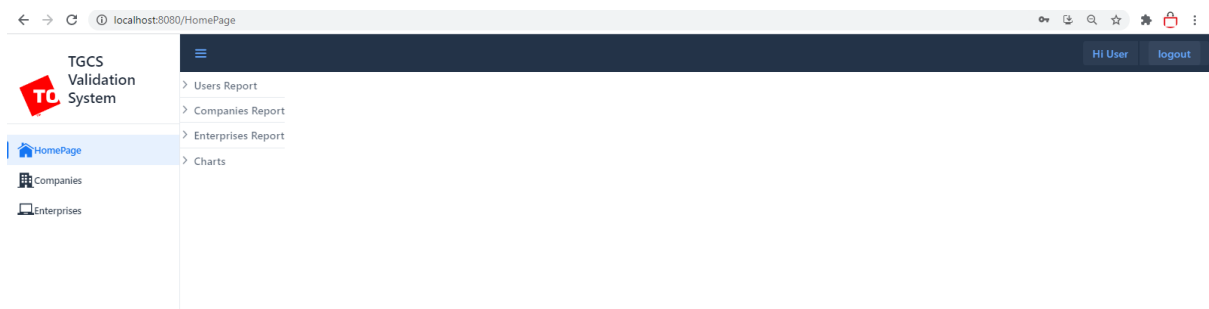


Figure 9. Hiding Users Tab for a user with role "USER"

```
54 @Route(value = "Users", layout = MainView.class)
55 @PageTitle(Constants.TITLE_USERS)
56 @CssImport("./styles/views/users/users-view.css")
57 @Secured(Role.ADMIN)
58 @PreserveOnRefresh
59 public class UsersView extends VerticalLayout {
```

Figure 10. Annotation used to secure a page for "Admin" role

3.3.3 Assigning password and password encryption

During User registration flow, field Binding rules validation has been considered for different fields, as you see in the figure below. For the password field, minimum password strength is enforced as a mixture of capital, small letters, numbers, and characters.

The image shows a user registration form with the following fields and validation rules:

- First Name:** Input field with a red error message "must not be empty".
- Last Name:** Input field with a red error message "must not be empty".
- Email:** Input field with a red error message "must not be empty".
- Password:** Input field with a red error message "need 6 or more chars, mixing digits, lowercase and uppercase letters". It includes a toggle for visibility (eye icon).
- Role:** Dropdown menu with a red error message "must not be empty".
- Activation Status:** A checkbox labeled "Activation Status".

At the bottom of the form are three buttons: "Save" (blue), "Cancel" (light blue), and "Delete" (light blue).

Figure 11. Binding Rules for User registration

But this is not how we are going to save the password in the database. With BCryptPasswordEncoder importing from Spring Framework Library, we can encode the password before saving it to the database. BCrypt is a password hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher. [4] BCrypt uses an adaptive hash algorithm to store passwords. BCrypt internally generates a random salt while encoding passwords, and hence it is obvious to get different encoded results for the same string. But one common thing is that every time it generates a String of length 60. This will allow us to increase the security of passwords saved in the database through any attack on the database. Of course, while in saving user flow encryption method is working, in the login flow decryption of password and a matching system is necessary which will be explained in next steps. [5]

3.3.4 Login overlay for VSCLS

To finalize the security for the application, we need a login overlay that is able to give access to the application only for those registered users in the User table of the database with saved email as username and password. This is done by Spring Security and overlay done by Vaadin Framework. The reverse of password encoding is working here, and a matching

system will allow registered users to access the application. Some functions which are done through Rest API will also be denied accessing the application if they are not permitted in the security class of Java. By listing the permissions in the security class, the listed functionalities will be able to pass through spring security. The inverse of login flow, logout flow, is also performed by a button inside the application. This button will route the user to the login overlay view again.

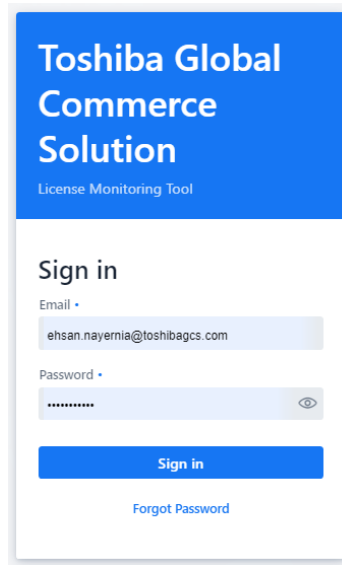


Figure 12. Login overlay

3.3.5 Forgot and Reset password flow

Each user can reset the password by following the forgot password flow enabled on the login overlay. Fortis functionalities while user inputting his or her email (username) application try to find it in the database and if it is available a token will be specified to that user in the table. This token is based on UUID, and it is almost with a reasonable unicity probability. A universally unique identifier (UUID) is a 128-bit number used to identify information in computer systems. The term globally unique identifier (GUID) is also used, typically in software created by Microsoft. According to the standard methods of generation of UUID, it

is unique for practical purposes. So, we can be sure if two users are trying to reset the password, the same token will not be generated for both of them. Since the token is generated, an email including a URL based on that token will be sent to the user's email (email and SMTP construction will be explained in the next topics). If the URL containing the token has the right token, the user will be able to change the password and save a new one by clicking on the URL and following the flow. Otherwise, if the token is not true, the matching system of the token will alert the user that you are not able to change your password. This is the way that we can be sure the right user who has followed the forgot password flow is now changing the password, not anybody else. After changing the password, the token value will be set to null value immediately for security issues and not to be used twice or by somebody else.

TOSHIBA VISUAL STORE

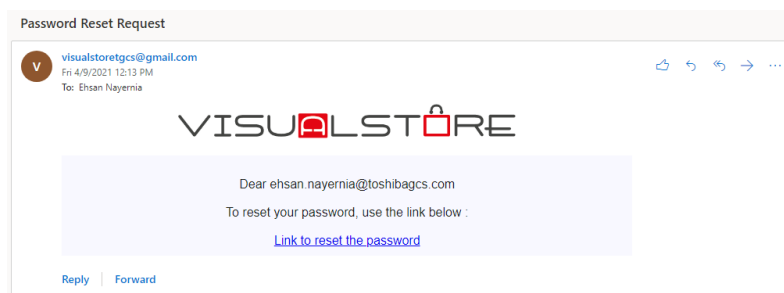
Toshiba Global Commerce Solution

Forgot Password

[Back To Login](#)

Input your email

[Send me recovery email](#)



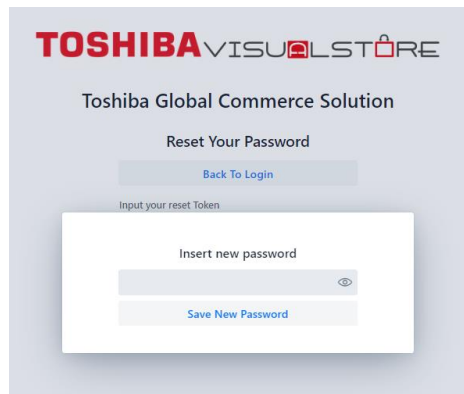


Figure 13. Steps of forgot and reset password flow

3.3.6 Keep tracking of Logged-in user as Current user

Tracking the logged-in user has provided some benefits for the application. First of all, in the layout, the user can see its name on the top right of the application. The second benefit is that for all changes in different tables, we can save a column in the related table to see who has created that record or last modifications have been placed by whom. For this purpose, only tracking of the logged-in user is not enough. We need to make a logical relation between the User table and the secondary table, for example, Enterprise or Company. The relation would be in kind of One-to-many, as explained before in database explanations. So, whenever we want to save a record for the enterprise table or Company table, the current User id will be saved. Thanks to the Vaadin grid component, we can perform the user's name instead of its Id inside the visualization of Grid on UI.

Modify	Name	Partita Iva	Created On	State	City	Province	Address	Created by
	Esselunga	123456789	2021-04-08	Afghanistan	Herat	s	s	Stefano Ales
	Coop	123654987	2021-04-08	Italy	Milano	Milano	Via x, 1	Ehsan Nayernia
	Lidl	123987456	2021-04-08	Italy	Milano	Milano	Via y, 2	Ehsan Nayernia
	Iper	951753852654	2021-04-08	Italy	Rome	Rome	Via Z, 3	Stefano Ales
	Aldi	951852357456	2021-04-09	Germany	Aachen	Aachen	xxx	Ehsan Nayernia

Figure 14. Saving User id of the creator in Company table and showing with the name in the visual grid.

```

20 @Entity
21 @Table( name ="company")
22 public class Company extends AbstractEntity {
23
24     @ManyToOne(fetch = FetchType.EAGER, optional = true)
25     @JoinColumn(name = "creator_id", nullable = true)
26     private User users;

```

Figure 15. Example of developing a Many-to-one relation between Company table and User table

3.4 SMTP implementations

The Simple Mail Transfer Protocol (SMTP) is stand for internet standard communication protocol for electronic mail transmission. With Java mail sender utilities of Spring boot and an SMTP connection made between VSCLS and a Gmail server account, VSCLS is able to prepare texts and components for sending an email in different situations and necessary occasions. For example, when Admin registers a user, an Automatic email will be sent to the registered email as username, and the user will be notified that his or her account is ready to be used with the preassigned password.

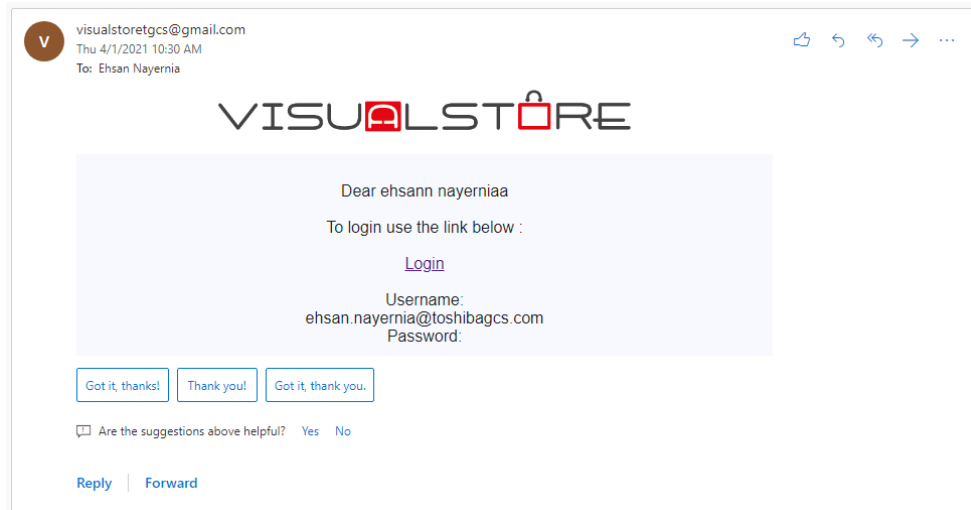


Figure 16. Sample Email sent via application and SMTP connection with Gmail account.

```

18 @Service("emailService")
19 public class EmailServiceImpl implements EmailService {
20
21     @Autowired
22     private JavaMailSender mailSender;
23
24     @Async
25     public void sendEmail(SimpleMailMessage email) {
26         mailSender.send(email);
27     }
28

```

Figure 17. How to start Java mail sender utilities

3.5 Reports in PDF and CSV

While VSCLS is a RESTful application (explained in theory part of REST API), all the PDF and CSV files will be ready to download after adjusting the layout through the GET method (`@GetMapping("/users/export/pdf")`) from a sub address of the application. These functionalities are done by annotation `@RestController` through a controller class. In this specific case, the Get method would be ready to download PDF files from `http://localhost:8080/users/export/pdf`. By a Search Field developed for each Table and Grid, the user can have a list of searched records in the table and bypass their ID as a parameter to the GET method; it is possible to download the PDF from only searched records of pdf. All

the searching fields in UI use the Query method on the database to find the searched terms inside the related table like the company table.

VISUALSTORE

List of Companies

ID	Company	PartitalVA	Created on	State	City	Province	Address
614	A	123456789	2021-04-13	Iran	Shiraz	a	a
575	B	123456789	2021-04-08	Iran	Shiraz	b	b
593	C	951852357456	2021-04-09	Iran	Shiraz	c	c
583	D	951753852654	2021-04-08	Italy	Rome	d	d
581	E	123987456	2021-04-08	Italy	Milano	e	e
579	F	123654987	2021-04-08	Italy	Milano	f	f

Figure 18. PDF generation from Company Table

VISUALSTORE

List of Filtered Companies

ID	Company	PartitalVA	Created on	State	City	Province	Address
583	D	951753852654	2021-04-08	Italy	Rome	d	d
581	E	123987456	2021-04-08	Italy	Milano	e	e
579	F	123654987	2021-04-08	Italy	Milano	f	f

Figure 19. example of PDF generation from Company Table by searching companies by their country "Italy"

```

12 public interface CompanyRepository extends JpaRepository<Company, Integer> {
13
14     @Query("select c from Company c " +
15           "where lower(c.companyName) like lower(concat('%', :searchTerm, '%')) " +
16           "or lower(c.city) like lower(concat('%', :searchTerm, '%'))" +
17           " or lower(c.state) like lower(concat('%', :searchTerm, '%')) " )
18     List<Company> search(@Param("searchTerm") String searchTerm);

```

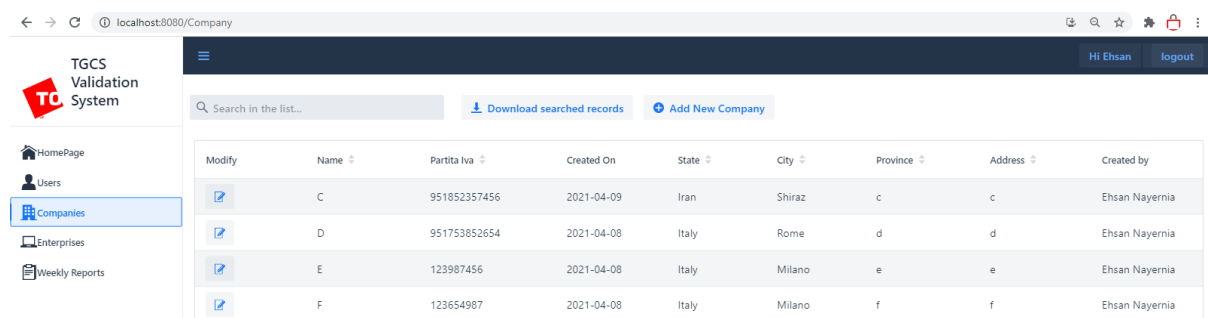
Figure 20. How to make a query on a table

CHAPTER FOUR

4 Secondary Implementations

4.1 Implementing company and enterprise table

The starting point of communications is to define a Company that wants to be in touch with VSCLS in the future. This action will take place by internal users of Toshiba, which has already been explained in primary implementations. Company Is a table in the database which has a relation with the user table to indicate the user who is going to make each record of company tables. By considering this logic, a One-To-Many relation could be deployed between User and Company Table (Explained in primary implementations). It means each user can make many rows in the Company table and be their creator but, each row of the Company table could be made by only one user. Columns of this table are the details of each company. This table could be more precise in detail in the future, but up to now, these Columns could make a reasonable sense of what we want to do in this application. The columns are shown in figure 20:



Modify	Name	Partita Iva	Created On	State	City	Province	Address	Created by
	C	951852357456	2021-04-09	Iran	Shiraz	c	c	Ehsan Nayernia
	D	951753852654	2021-04-08	Italy	Rome	d	d	Ehsan Nayernia
	E	123987456	2021-04-08	Italy	Milano	e	e	Ehsan Nayernia
	F	123654987	2021-04-08	Italy	Milano	f	f	Ehsan Nayernia

Figure 21. Columns of Company table

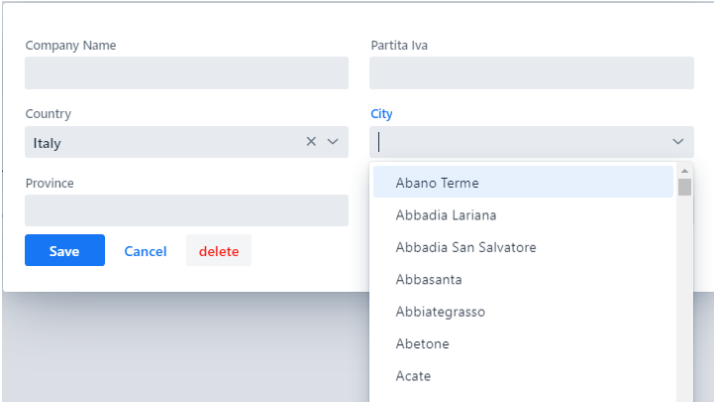
One of the Usages of Rest API is here with a Post method from an external server for the list of Countries and Cities. To have a better experience for the user, country and city could be chosen among a list which has been provided by a Post method from an external URI. It was

also possible to make a database of countries and cities manually, but with REST API for sure, it is much more efficient, faster, and more reliable. The sample code is showed below in figure 21 for this purpose, and the result is obvious in the figure below.

```
String url = "https://countriesnow.space/api/v0.1/countries/cities";
URL obj = new URL(url);
URLConnection con = (URLConnection) obj.openConnection();

// Setting basic post request
con.setRequestMethod("POST");
con.setRequestProperty("User-Agent", "MyApp");
con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
con.setRequestProperty("Content-Type", "application/json");
```

Figure 22. Sample Code for Listing cities with Post Method



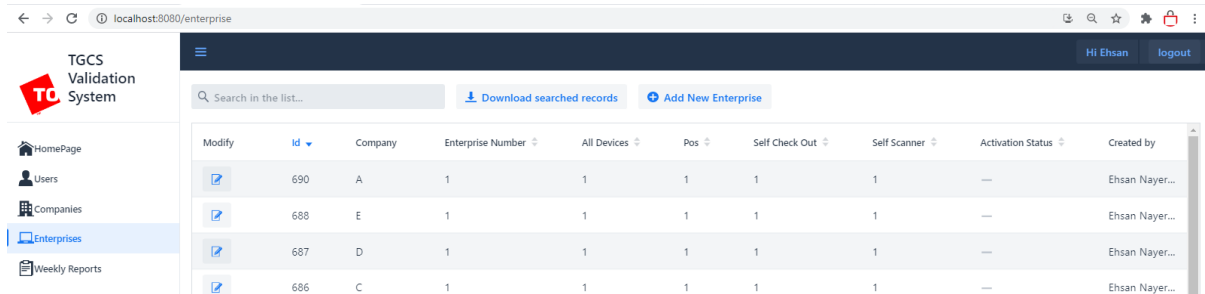
The image shows a web form for creating a company. It has several input fields: 'Company Name', 'Partita Iva', 'Country' (with 'Italy' selected), and 'Province'. A 'City' dropdown menu is open, displaying a list of cities: Abano Terme, Abbadia Lariana, Abbadia San Salvatore, Abbasanta, Abbiategrosso, Abetone, and Acate. At the bottom of the form, there are three buttons: 'Save' (blue), 'Cancel' (light blue), and 'delete' (red).

Figure 23. User Interface of Company Creation flow.

The next necessary table that must be created is Enterprise Company. Basically, each company can have a single enterprise system that must send the reports to VSCLS. So, by considering this logic, a One-to-one Relation could be deployed between these two tables (explained in primary implementations). It means the row of the Company table identified by its Id could be in relation with only one row of Enterprise table identified by an id and vice versa.

Like what we did in the Company table, a One-To-Many relation could be considered between the User and Enterprise table to determine the creator of each record. For instance,

the columns of the Enterprise table which can make a good sense of this application are indicated below:



Modify	id	Company	Enterprise Number	All Devices	Pos	Self Check Out	Self Scanner	Activation Status	Created by
	690	A	1	1	1	1	1	—	Ehsan Nayer...
	688	E	1	1	1	1	1	—	Ehsan Nayer...
	687	D	1	1	1	1	1	—	Ehsan Nayer...
	686	C	1	1	1	1	1	—	Ehsan Nayer...

Figure 24. Columns of Enterprise table

In the creation of each enterprise, there are some data related to the number of subdevices working under an enterprise of the company, but there is two important Column that could be a key to the next features.

- Activation Code
- Activation Status

When an internal user of Toshiba wants to create a new enterprise, a random unique Activation code will be dedicated to each enterprise of a company, and the activation status of the enterprise will be set to a default value of false, which means that the enterprise is not activated yet and must be activated through the first installation. At the end of enterprise creation, a PDF containing the activation code will be ready to download like the figure below, and this pdf will be sent to the company manually by an internal user to be used during the first installation flow. This part will be explained in the next part.

As explained in Reports in PDF and CSV, downloading a PDF containing enterprise detail from VSCLS showed in figure 24 has also used the Get method of REST API.

Customer Name
A
Partita IVA
1
Enterprise Activation Code
9e4084ce-42d3-4e1f-bb78-757a6caff334
Activation Status
Not Active
Created On
2021-04-15T11:56:02.322904
Updated On
2021-04-15T11:56:02.322904

Figure 25. Enterprise detail including activation code in PDF

This point must be considered all the flow that has been explained during the implementation of company and enterprise, was a manual flow that an internal user of Toshiba can follow. Since the number of companies and number of licenses that customers will buy for sub-devices will be so much, another approach could also be considered. All the licenses will be bought through Oracle, and a CSV file including all the information will be provided to Toshiba. An Automatic system inside VSCLS that can receive this information from CSV files, parse them, and put them inside the related columns of the company table and enterprise table could be an appropriate solution. This procedure could also be managed by REST API to upload the CSV file on the VSCLS. If the company does not exist, it could be a Post method to make a company and also its enterprise with the number of sub-devices. If a Company exists and only the enterprise wants to be changed, a Put method is enough to update the company's details and its enterprise.

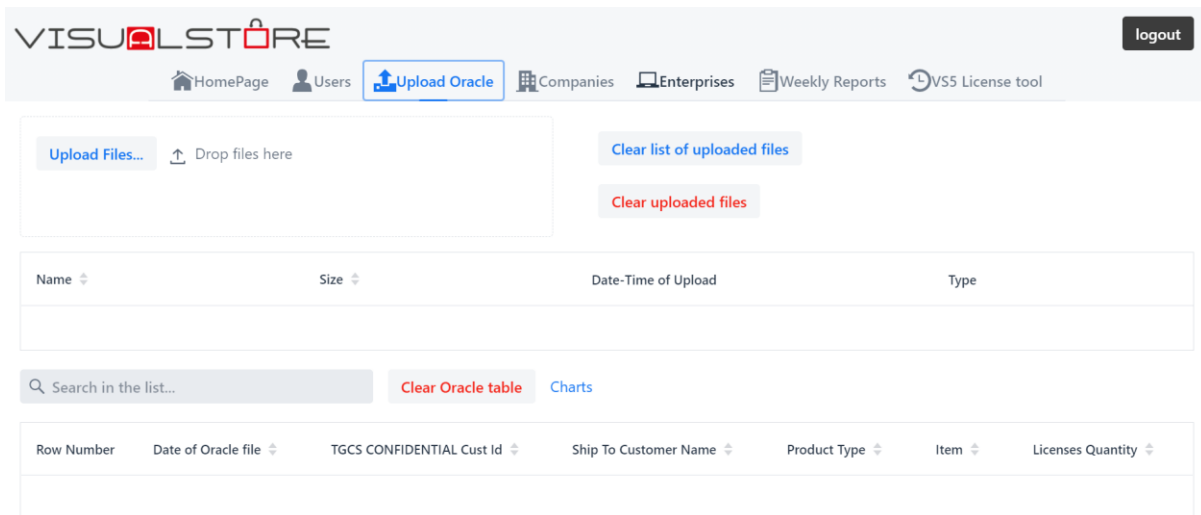


Figure 26. UI for uploading a CSV containing the Companies information

4.2 First installation of VS on enterprise and activation flow

For the first installation of VS on the enterprise of the company, a module inside the enterprise saves the activation Code, and after the first installation, the enterprise must make a connection with VSCLS to send a Json message through REST API with put method to activate its status in the database. The flow below indicates this procedure. This must be noted that the activation code is unique, and each company has its own activation code sent to the company after Enterprise creation by an internal user of Toshiba.

When the JSON message is sent to the VSCLS, three events can be happening.

- Event 1: If the activation code in JSON message is equal to an activation code in the database and the related activation status in the database is false, the activation status would become true (active) with the put method of REST API, and 200 Ok messages will be sent as a response to the enterprise.
- Event 2: If the activation code in the JSON message is equal to an activation code in the database and the related activation status in the database is true, the activation status would remain true (active), and 208 208 already Reported message will be sent as a response to the enterprise and indicates that this enterprise is already active

- Event 3: If the activation code in JSON message could not be found with the matching system in the database, no activation procedure will be processed, and a 404 Not found message will be sent to the enterprise, which asks the enterprise to send the right activation code for activation procedure.

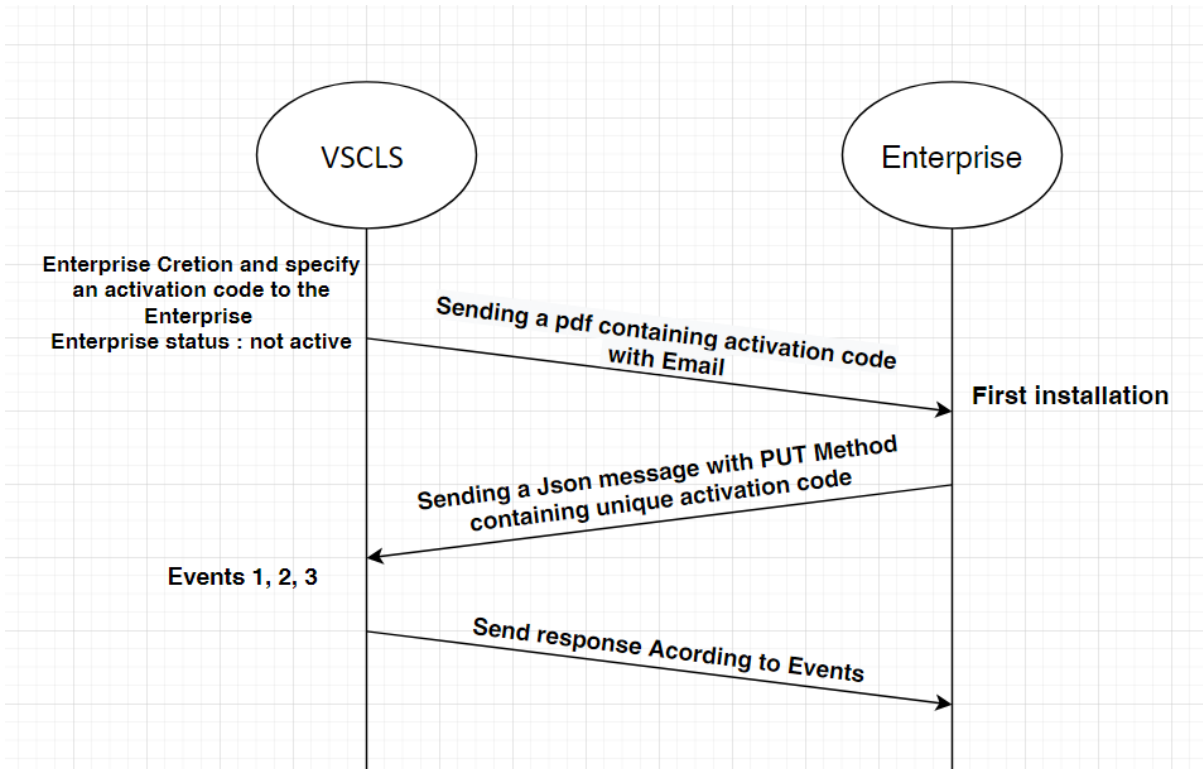


Figure 27. First Installation and activation flow of enterprise

For testing the application and REST API procedures, a test JSON message will be sent through applications like Postman or Advanced REST client to the sub address of the application. Here is a sample of JSON message sending from the enterprise to the application. With mapping annotation, the address <http://localhost:8080/enterprise/activate> is responsible for receiving and analyzing the put method coming from the enterprise. The figure below is an example of a JSON message from the enterprise and simulated by the Advanced REST client.

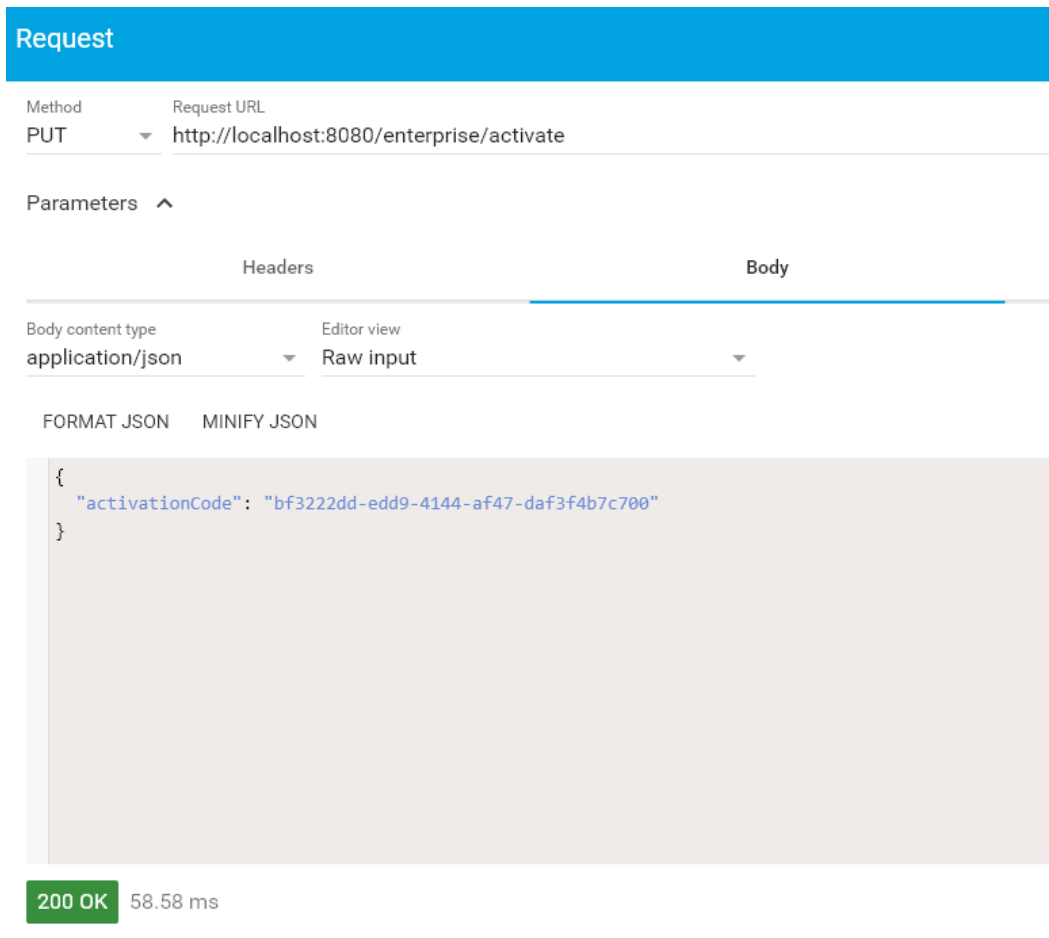


Figure 28. JSON message from enterprise to VSCLS for activation flow and the response of 200OK

4.3 Weekly report

When the enterprise becomes an activated one, it is time for sending weekly reports. The first weekly report coming from the enterprise is critical because enterprise must first be activated; otherwise, it is not valid to send weekly reports. The hardware serial number of the enterprise must be contained in weekly reports, and the hardware serial number mentioned in the first report is a comparison reference for checking the next weekly reports. The following weekly reports must be sent via the same device with the same hardware serial number of first weekly report; otherwise, a visual warning will be shown to the user of VSCLS. Time stamping of all weekly reports coming from enterprises is a helpful issue to track the unauthorized reports and find the first weekly report to make it a reference for the next

weekly reports.

In the enterprise table explained in the previews section, all number of devices working under enterprise was completely declared by a user of VSCLS or by uploading a CSV file of Oracle containing the number of purchased licenses for sub-device. It is also another reference for making comparisons and stops unauthorized enterprises. The total number of devices declared in each weekly report must be equal to the total number of devices declared in the enterprise table; otherwise, a visual warning will be shown on VSCLS.

It is obvious that weekly report table and enterprise table are also in relation of One-To-Many. It means that each enterprise can have several weekly reports, but this relationship could be based on something else instead of primary key identification, which VSCLS users only know. Yes, in enterprise table, we have another unique constraint, and that is activation code. Including activation code in every weekly report can be a good unique identifier for making a relation between weekly report table and enterprise table.

Like before, the request format coming from enterprise is a JSON message with the PUT method in REST API.

The weekly report coming from the enterprise will be stored in weekly report table, even if the weekly report is authorized or unauthorized. Taking action against unauthorized weekly reports must be decided according to the needs of Toshiba and policies.

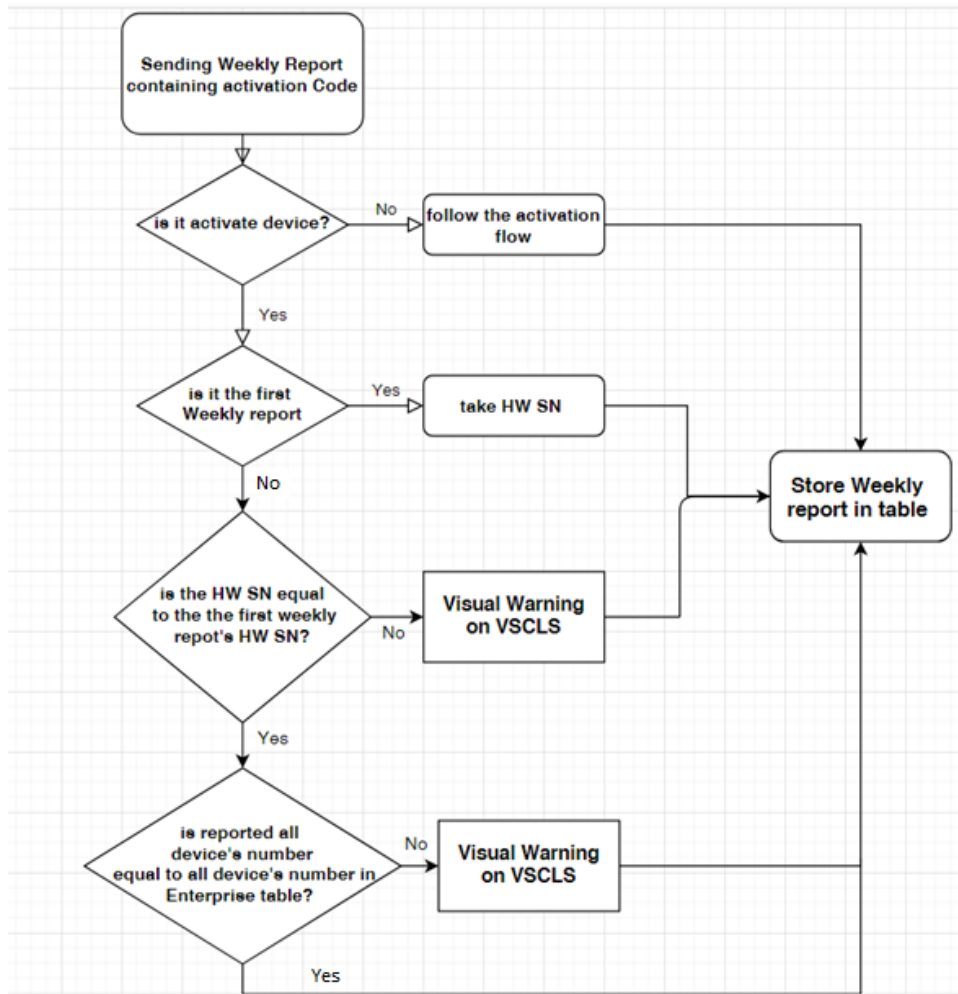


Figure 29. Weekly report flow

CHAPTER FIVE

5 Performance evaluation

Now the skeleton of our application is ready. During the two main implementation parts (primary and secondary), I tried to show more or less the result of developing and test them to make the explanations clearer. But before concluding the article, checking the performance and see if the proposal is working in all aspects is necessary.

This application will cooperate with the main core of software; VisualStore. So both VisualStore version 6 and the new version of license monitoring toll VSCLS must be fully developed and ready to be functionalized in production mode. But before production, VSCLS must be tested entirely in the laboratory of Toshiba to be sure it is working correctly as a license monitoring tool and the solution has a good performance against the proposal's anomalies. But even earlier, before testing in the laboratory, some tools can give us the ability to test VSCLS and give us a precise performance evaluation. Since many main duties of VSCLS are done with REST, we can simulate our tests and see the result of the connection between VSCLS and simulated enterprise, which is installed with VisualStore. Postman is a platform that can make these simulations for our performance evaluation.

To quickly summarize what we did during the secondary implementation, we have defined Companies, enterprises. The enterprise must be first activated. A module inside the core of VisualStore will send a PUT request to VSCLS via REST to ask for activation. The activation code has already been sent to the company in the creation of the company. So the enterprise must be able to ask for activation with its activation code. This result can be seen in the figure below and the different responses of VSCLS according to the different situations of enterprise in the database.

The screenshot shows a REST client interface for a PUT request to `http://localhost:8080/enterprise/activate`. The request body is a JSON object: `{ "activationCode": "1cac67f7-87b4-4f05-9ea1-070688d45303" }`. The response status is `200 OK` with a response time of `44 ms` and a size of `1.32 KB`. The response body is empty.

Figure 30. 200 Ok Response to an existing enterprise with valid activation code.

The screenshot shows a REST client interface for a PUT request to `http://localhost:8080/enterprise/activate`. The request body is a JSON object: `{ "activationCode": "1cac67f7-87b4-4f05-9ea1-070688d45303" }`. The response status is `208 Already Reported (WebDAV) (RFC 5842)` with a response time of `26 ms` and a size of `339 B`. The response body is empty.

Figure 31. 208 already Reported response to an already activated enterprise.

The screenshot shows a REST client interface for a PUT request to `http://localhost:8080/enterprise/activate`. The request body is a JSON object: `{ "activationCode": "EXAMPLE of incorrect activation code to receive 404 not found response" }`. The response status is `404 Not Found` with a response time of `17 ms` and a size of `1.1 KB`. The response body is empty.

Figure 32. 404 Not found response to an activation code which is not existed in database

In continuing the flow of activation, when an enterprise is activated, it has the ability to send weekly reports. Again, this test can be done with the Postman simulator of REST.

Here in the figure below, the visual warnings can be seen against different kinds of anomalies explained during the article.

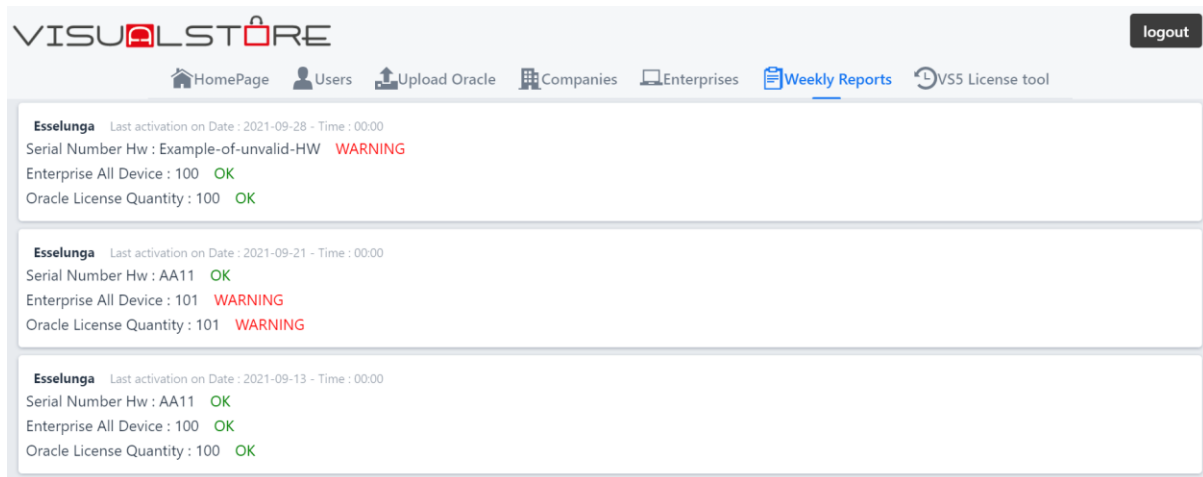


Figure 33. Visual warning against anomalies during three consecutive weeks

This point must be considered that all the data coming from an enterprise as a weekly report will be saved in the database properly. This data can be used in the future to take appropriate action for unauthorized enterprises which have anomalies and violations of license rights.

Up to now, the performance of the solution has been evaluated. In the development steps, the application is responding to the REST requests and database correctly. Also, well cooperation can be seen among the technologies that have been used during the development. Thanks to Java, Spring framework, Vaadin framework, PostgreSQL database, a reasonable relation has been developed between the backend and frontend of the application. REST communications and sending emails via SMTP are also acting on time with a low delay that can be neglected.

CHAPTER SIX

6 Conclusion and future works

Up to now, we have succeeded in developing a RESTful application, “VSCLS,” whose main duty is to check and control the licenses and number of valid devices working with an enterprise that is also installed with VisualStore application. VisualStore is the software produced by Toshiba Global Commerce solution Italy, a POS platform for any kind of hyper stores.

Each company that is a customer of Toshiba can have an enterprise that controls all the activities of stores and active sub-devices. This enterprise must be in touch with VSCLS and report its validity and sub-devices to VSCLS. A module inside the core of VisualStore installed on the enterprise asks for the validity and license of each sub-device wanted to be connected to the enterprise. Enterprise will collect all the reports coming from all devices and send them all to VSCLS. Since Toshiba has decided to immigrate from version 5 to 6, this module must be designed and set inside the VisualStore.

As we have seen during Implementations, VSCLS, as a server inside Toshiba, has this ability to perform connection with the client-side, which is the enterprise of companies. All the connections are through the web area, and they are using REST API architecture. Database of VSCLS is designed with PostgreSQL, which mainly contains the data of users of TGCS, which are internal employees of Toshiba, data of Companies, their enterprises, and licenses, and the number of active sub-devices.

The main duty of VSCLS could be seen in the first installation of enterprises and weekly reports. Weekly reports are a set of rules and communication protocols between enterprise and VSCLS for reporting the validity of licenses and the number of sub-devices to VSCLS. All the reports use REST architecture to make communication between enterprise and VSCLS. Generally, a weekly report can have two main news. One is that enterprise and sub-

devices are working properly according to the number and kind of licenses, or there is a problem. This problem could be that the hardware serial number of the device which is sending the report has been changed according to the first weekly report had been come from enterprise, or the number of sub-devices in complete detail of kinds, are more than the number of licenses which are available inside the database of VSCLS, or maybe any cloning of software and licenses can happen on the subdevices. If weekly report wants to report any anomaly before going to be saved in the database for future traces, it makes a visual warning on the system of VSCLS. This is the first step to have complete information about enterprise and subdevices and the status of their licenses. But as it could be understood from the name of the license monitoring tool, it is not enough to have only a visual warning against anomalies. VSCLS must have the ability to stop anomalies and perform an action against any violation of license rights. This could be considered as a future point that must be developed by Toshiba team. There are many proposals against policy violations. Toshiba must decide what the best action that can be taken in this situation is. For sure there are many considerations to make this decision. From an economic point of view, this action must be in a way to have the best result both for customers and Toshiba. The strictest approach that can be performed against any anomalies is that change the enterprise activity status to inactive, but we must see acting like this is reasonable or not. Anyway, as the next starting point of development, this topic could be a good step: Rules against unauthorized enterprises.

Implementing the JWT method is another proposal for future works. But what JWT is? JWT stands for JSON web tokens. Since all the transmitting information between parties (client and server) are in JSON format, JWT defines a standard that provides a secure way of transmission. The information in JSON format will be digitally signed so it can be verified and authenticated. Like every other security method, in the JWT method, also we can use two general methods of private(secret) key or public(general) key. The structure of JWT method

has three main elements of Header, Payload, and Signature. In the header, it will be explained the method, which is typically JWT, and the algorithm used in signature (private or public algorithms like HMACK or RSA). Payload is obviously containing the information that wants to be transmitted between server and client in JSON format. Finally, in the signature part, you need to have a mixture of header, payload, and algorithm of the method to sign and encode the information.

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoiYWRtaW4iOnRydWUsImVudCI6MTUxNjIzOTYyMn0.bQTnz6AuMJvmXXQsVPrxeQNVzDkimo7VNxxHeSBfClLufmCVZRuuyTwJF311JHuh
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
<pre>{ "alg": "HS384", "typ": "JWT" }</pre>
PAYLOAD: DATA
<pre>{ "sub": "1234567890", "name": "John Doe", "admin": true, "iat": 1516239022 }</pre>
VERIFY SIGNATURE
<pre>HMACSHA384(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-384-bit-secret) <input type="checkbox"/> secret base64 encoded</pre>

Figure 34. example of JWT method [6]

REFERENCES

- [1] K. Lange, THE LITTLE BOOK ON REST SERVICES, Copenhagen, 2016.
- [2] R. Fielding, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Internet Engineering Task Force (IETF) , 2014.
- [3] R. T. Fielding, "Representational State Transfer (REST)," in *Architectural Styles and the Design of Network-based Software Architectures*, UNIVERSITY OF CALIFORNIA, IRVINE, 2000.
- [4] N. Provos, D. Mazières and T. J. Sutton, "A Future-Adaptable Password Scheme," in *USENIX Annual Technical Conference*, 2012.
- [5] D. Rai, "Storing Hashed Password to Database in Java(Using Bcrypt)," [Online]. Available: <https://www.devglan.com/spring-mvc/storing-hashed-password-database-java>.
- [6] "JWT," [Online]. Available: <https://jwt.io/>.
- [7] F. S. Ivan Salvadori, "A Maturity Model for Semantic RESTful Web APIs," in *Web Services (ICWS), 2015 IEEE International Conference*, New York - USA, JUNE 2015.
- [8] L. Richardson and M. Amundsen, RESTful Web APIs, O'Reilly Media, 2013.
- [9] M. Jones, JSON Web Token (JWT), Internet Engineering Task Force (IETF) , May 2015.