Trains energy meter system using Blockchain

Author: Duch Jonama, Gerard
Director : Desbat, Stephane
Internal examiner: Teniente, Ernest
Company: Sogeti High Tech
Bachelor Degree in Computing Engineering
Speciality: Software Engineering
Barcelona School of Informatics
Universitat Politècnica de Catalunya - BarcelonaTech (UPC)
7 October 2017

INDEX

1. ABSTRACT

Every day, there are more people who start to be interested in Blockchain technology and their applications, in consequence, it has gained a lot of importance.

In this paper, I will present a solution to a problem that the France rail network will have in a near future. Currently, they are being used for the French public railway company, but in a not-too-distant future, they will let to some other companies use them.

As a consequence of this change, they will have to make some changes at the management level, for example, consider how to divide the rail network maintenance costs, calculate how much energy consumes each company, etc. Nevertheless, at this paper we have focused exclusively to develop an application that will find out the consumed energy by each company and, due to the fact that this project will involve more than one company sharing sensitive data, we also have to ensure the security and transparency of it. This is the reason why we have chosen to develop it using the Hyperledger Fabric Blockchain implementation, and profit its architecture benefits.

To sum up, we present a simulator who randomly generates train energy data who at the same time are stored in two types of databases, Blockchain and NoSQL. Finally, we have a client who allow to every company user to manage the energy consumed that his company trains has wasted.

2. PREFACE

The reason I started this project came at the end of my exchange studies in Lyon, France, in January of 2017.

As I I just has left the final project of my Bachelor Degree in Informatics Engineering to finish my university studies, and after having an interesting, revealing time, I realized that I wanted to stay for a longer time at Lyon. So, I decided to search for a company who offered an interesting internship and do my final degree project with them.

In order to achieve this objective, and after an exhaustive research and effort, I found an opportunity I could not pass up. I applied to a Blockchain and Internet of Things internship at Sogeti High Tech, a Capgemini filial. After a personal and competence interview, I was one of the two students chosen for this internship in Lyon.

My main goal in this internship was to learn the new Blockchain world as well as gain work experience and continue improving my French. And finally, when I just had one month left to leave, I realized that I did not only had hardly achieved my objective, I also learned a lot of teamwork skills and an undefined time contract.

Overall, I can conclude that after having a really good internship time and team, I had decided to stay with that company for a longer time and accept the contract. As it offers me a good future and the possibility to continue improving my academic and professional skills.

3. INTRODUCTION

This project is done using an agreement of educational cooperation between the Barcelona School of Informatics and Sogeti High Tech [1].

Sogeti is a leader in local-sourced IT and engineering services, specialising in application and infrastructure management and technology consulting. It offers innovative solutions for Testing, Business Intelligence Management, Mobility, the Cloud and Cybersecurity, supported by its Rightshore global service model and its methodology. With a presence in 15 countries and over 100 branches in Europe, the US and India, the company boasts a workforce of 20,000 professionals. Sogeti is a 100%-subsidiary of Cap Gemini S.A. [2], listed on the Paris stock exchange.

3.1 The problem

In a competitive environment with a global and changing market, companies must be able to gain a competitive advantage over their rivals, mainly based on their growth and customers' loyalty. So, the multinationals companies, as Sogeti High Tech, needs to have a big set of clients who have confidence to them, in order to have lots of new projects.

Due to the fact that Sogeti High Tech has already made a big amount of projects for the most important rail companies from France, as SNCF [3] or Alstom [4], we can say that Sogeti High Tech has a deep experience at the railway field, among many others. This experience allows them to have a future vision of the French rail network and the improvements or changes that will occur to them. At this point, is where we find a new problem that SNCF will have in a not-too-distant future. Currently, the French rail networks are just used by the French public railway company SNCF but, in not a long time, they will let some other companies use them, for example, the trains from the Spanish public railway company Renfe [5].

As a consequence of this change, they will have to make some changes at the management level, for example, consider how to divide the rail network maintenance costs, calculate how much energy consumes each company, etc. And Sogeti High Tech, wants to be the company who implements the necessary solutions to allow the correct cooperation of every actor.

As we can imagine, every change involves an exhaustive and difficult analysis of the different scenarios, so, that is why this project will be focused exclusively on the energy issue, in other words, find out the amount of energy consumed by each company. Then, some other teams of the

company will take care of the other parts of the future system, so as to build the solution who will fit to the future requirements.

Finally, a part of the main problem, our solution has to ensure the data security and transparency, due to the fact that this it will involve a big sort of companies and a lot of money, so this will be an additional difficulty that we will have to face.

3.2 Objective

After seeing the problems that we want to solve in this project, we have defined some objectives that will ensure the solution of them. Firstly, the principal objective is to develop an application that allows the clients to manage their trains energy waste, in safe, transparent and accurate way. So, in order to achieve the main purpose, I have defined some other goals that if we reach them, we can suppose that the first objective is accomplished.

Those other goals are:

- Manage the time that each task of the project is going to take in order to know as well as we can the waste of time.
- Manage the deadlines of the tasks that have to be done in order to have a lot of feedback.
- Divide the work to be done in little tasks in order to let us to manage separately.
- Divide and allocate the tasks to the different member of the development team.

As we can see, the majority of these objectives are based on data compilation and analysis. So as to find out this data, we have tracked every move done in the project development.

3.3 Stakeholders

As this project have been realised at Sogeti High Tech, we have had the following stakeholders:

Capgemini, Sogeti and Sogeti High Tech company

Capgemini, Sogeti High Tech and Sogeti are the same company, we have that Sogeti High Tech and Sogeti are filial from Capgemini.

In this project, the company Sogeti High Tech have develop the product. But the rest of the company has been interested on the project because they can profit from our experience or from the project, they may realize similar projects.

Project director

Stephane Desbat has been the project director, his role has been to guide and supervise the development team in order to achieve the main objective.

Development team

In this project, the development team has been formed by two developers, Terry Pasquet and Gerard Duch Jonama. Due to the lack of team members, we have adopted different roles during the project development at Sogeti High Tech.

The users

The application is destined to them. Particularly, to all these enterprises who has to share the energy waste with other enterprises.

Capgemini, Sogeti and Sogeti High Tech clients

As I said before, Sogeti High tech is direcly related to Capgemini and Sogeti, so it has a strong customer base. And, as it is the same company, all the clients from them could be a potential buyer.

3.4 The team

Currently, even the enterprise is huge and with a lot of employees, we are just three people working on the project. Each of them brings a different background and skills to complement the team.

Following I write a short background description of each member of the group.

Stephane Desbat
Chief of the project

Stephane is an experienced innovation consultant who is experienced in digital manufacturing about the internet of things and Big Data architecture, predictive maintenance and manufacturing intelligence issues.

Terry Pasquet
Future Software Engineer at Leonardo da Vinci Engineering School student

Terry is a French Engineering School student who is doing the final bachelor internship. He is the responsible for the Front-end and Blockchain part of the project, as he has experience in these fields.

Gerard Duch Jonama (author)
Future Software Engineer at Universitat Politècnica de Catalunya and BarcelonaTech

I am a young student from Barcelona who works as a software engineer. I am the server, simulator and database responsible.

3.5 State of art

As it is explained at the problem presentation, this project is oriented to the future of the French rail networks, as a result of this fact, we are going to build a new field solution.

Despite the number of rail networks that exists, it's difficult to find one example that has the same scenario that we will have in France. Due to this fact, the examples that I have considered to study are some projects who have similar requirements or characteristics to the ones from our project.

3.5.1 Similar projects

Intelligent Vehicle-Trust Point: Reward based Intelligent Vehicle Communication using Blockchain [6]

This is a paper who presents an Intelligent Vehicle-Trust Point (IV-TP) mechanism for Intelligent vehicle (IV) communication among IVs using Blockchain technology to achieve the data security and transparence that they need.

As we can see, this paper does not involve an architecture like the one that we need. Due to this fact, this project would not be a solution to our problem. However, they use the Blockchain in order to achieve a big data security and transparency. So, bearing in mind that we will work using sensitive data, we are interested to use this technology.

Blockchain enabled Trust & Transparency in supply chains [7]

This thesis' research purposes to investigate the nature of trust in supply chains and if Blockchain technologies can increase trust to them.

Considering the thesis' conclusions, Blockchain technology can be used to build open, secure and trusted systems assuming that the infrastructure processing and recording transactions is secure and properly managed. So it can increase the information transparency and tracking in supply chains.

Finally, as we had at the last similar project, we see that this wouldn't solve our requirements, as the assets that we will exchange are completely different. Nevertheless, we can have the same conclusion. We reiterate to use the emerging Blockchain technology, in order to achieve the transparence and security that we need.

Blockchain – an opportunity for energy producers and consumers? [8]

This is the unique paper who does not present a concrete project. They put under analysis the possibility to use of public and private Blockchain in the energy field. As we will change energy assets at our project, I have found interesting to cite this study.

Even though they say that it will minimize the costs and will increase the transparency, speed and flexibility, it concludes that Blockchain technology is still in its infancy, and that many experts also suspect that Blockchain technology might not be as scalable as needed.

3.5.2 Conclusion

After doing a deep search of the similar projects, we can conclude that no one has built a project like the one we want to realise.

In one hand, the only paper who approximates more to our field, is the one called "*Blockchain – an opportunity for energy producers and consumers*", and it is too old to accept their conclusions, as new Blockchain implementations has been emerged since they wrote them. So we can ensure that we will cover one market spot.

On the other hand, the other projects are different that the one we will develop. Nevertheless, we can use the Blockchain technology in order to achieve the security and transparency that our project also needs.

4. SCOPE

Due to the fact that this project will have a huge scale, I will concretely define the main functionalities that my final degree project will cover:

- Develop a software who will automatically simulate the energy consumed by the trains while they do their routes throw the French rails network.
- Develop a web service who will allow to the users to check the energy consumption of the trains.

4.1 Project main obstacles

The limitations and obstacles that I had during the project development were mainly caused by the lack of knowledge, as I we have worked in a completely new environment with some technology that I have never worked with before.

First of all, in order to simulate the data that the trains sensors send, I had to understand which kind of information they provide and learn how the French rail networks are built and work, so as to develop the simulator that will generate a coherent energy data. For example, we know that there are trams that the trains are powered by AC energy and others with DC, or there are trains that will consume more energy than others.

Secondly, we have developed the application, using two databases that I have not used before. Due to this fact, it has been complex to understand and develop an solution using them, as it has been difficult to find information.

Thirdly, we have used Python in order to develop the simulator. As it has been the first big project that I have developed using this language, It took some time to integrate myself into it, even though I had experience with other programming languages that are close to Python.

Finally, in this project I had an added difficulty due to the language. As i have done it in a French enterprise, the communication between the other members of the team is in French.

5. METHODOLOGY

There is a big sort of software development methodologies, from the traditional Waterfall to the complicated Rational Unified Process.

In order to choose one that fits to your project, you have to think about his context. As this project it is for a set of companies that we do not know, the requirements can not be perfectly defined, so we have added some changes while we have developed it. Apart from that, we have done presentations and demonstrations in order to have feedback and add value to our application.

As a result of these characteristics, we have thought that the Agile methodology [9] would fit perfectly to us. As the agile manifesto [19] says that the individuals and interactions are over the processes and tools, working in software is over the comprehensive documentation, the customer collaboration is over the contract negotiation and responding to change is over following a plan.

5.1 Agiles methodologies implementations

After we had decided to use the Agile methodology, we thought which implementation will fit better to us. The most famous agile methodologies implementations are Kanban [11] and Scrum [12], so before we decide the one that we are going to apply, I am going to introduce them to yourselves.

Kanban

Kanban is a method for visualizing the workflow, in order to balance demand with available capacity and spot bottlenecks. Work items are visualized to give participants a view of progress and process, from start to finish where the team members pull work as capacity permits, rather than work being pushed into the process when requested. This provides a visual process management system which aids decision-making about what, when and how much to produce.

Scrum

Scrum is a management and control process that cuts through complexity to focus on building products that meet business needs. Using this agile implementation the management and teams are able to get their hands around the requirements and technologies, and deliver working products in an incrementally and empirically way.

Conclusion

After these short definitions, we can see that both of them could have perfectly fitted for us. However, if we go deeply, we can see that they have some differences that will help us to choose one.

In one hand, Kanban helps you with the processes that are already working, trying to improve them over time, without shaking up the whole system.

While in the other hand, Scrum can help you to organise and structure your work in an efficient way, with a big flexibility and adaptability for the changes and the requirements that are not perfectly defined.

As we knew, our project would be adapted while we got the feedback, that is why we though that the Scrum implementation would be the best implementation for us, as it is the one who gives us a better adaptability and flexibility.

5.2 The process

The Scrum starts with the different iterations or sprints definition. These iterations have to be defined at the temporal planning and are going to be attached to some user history that should be done before the sprint ends. Each user history is a concrete and completely independent functionality with an estimated level of difficulty. At this project, in order to estimate that, the development team have met and determined a temporal period to each one, instead of using the history points.

As in every iteration we will have some functionalities done, at the end of the sprint, we will have added value to the software, and maybe, a built product. Therefore, we can do a demonstration to the users or clients in order to get some feedback. In our project, the development team have done this work.

At the end of each iteration, the team have to meet so as to assess the feedback received and may add some new user histories depending of it.

Finally, we have followed this process using a special “to-do list” software, where we just write the name, description and amount of time that every task of the Sprint will take.

5.3 Validation

As in every Agile project that is implemented using Scrum, each user history has to be followed by some tests that will verify his own good execution. In order to prove this, we have executed some

unit tests that will certify the correct execution of the whole program. And, as a result of this execution, the other members of the team have accepted or declined the user history correctness.

So, in order to check the correct execution of our project, we have created unit tests for every module of the project that have been executed every time that we have made changes. Finally, we have used the tests outputs in order to verify the code execution.

5.4 Development tools

In order to explain the development tools, I will divide the them in some big groups, as we have used so many different technologies and tools in order to develop our project. Concretely, there are four groups, first I will introduce the tools that we have used at the whole project, then the tools that we have used to develop the Simulator, thirdly, I will present the tools that we have used to develop the APIs and the Front End, and finally, the ones that we have used to locally deploy the Blockchain.

So, the tools that we have used in every module of the project are, the Apache Subversion [13], used to share, track and manage the work that is done. Then, so as to manage the SCRUM tasks, the project specification and the application repository, we have used the TeamForge as it enforces a short cycle approach, forces the developer to document the changes that he does and it allows you to see some project statics as well as is the unique tool that our enterprise provides us. After that, so as to develop the software, we have used Sublime text and VirtualBox, so as to have an Ubuntu machine and develop the modules using this environment, as the Debian based operating systems offers us more applications to deploy and develop software. In order to write the final degree project, I have used the Pages editing application. Finally, the communication between the other colleagues from the enterprise has been done using the Skype for enterprises and the Microsoft Outlook.

Referring to the tools used to develop the first module of the project, in other words, the Simulator, are basically the tools that we have said in the previous paragraph as well as Jenkins, the Framework who will help to us to check that everything is working correctly.

The tools used to develop the Cloudant and Blockchain APIs and the Front End are basically the IBM Bluemix, so as to deploy the APIs, the Client and the Cloudant database, and the Postman, so as to verify the correct execution of the Blockchain API and the Front End.

Finally, the tools used to deploy the Blockchain are the Docker technology, that has been used to create and run the images of the every element that we need to have a Blockchain, and the Rocket chat [14], so as to solve Hyperledger Fabric technical doubts.

6. PROJECT SPECIFICATION

So as to introduce the project requirements, I have divided them into two groups due to the connexion between them.

On one hand, we have the simulator. One module who runs independent from the rest of the software, as it will represent a running train.

On the other hand, we have the NoSQL and Blockchain databases, their respective APIs and the Client, who allows the users to manage the stored information at the two databases.

6.1 Simulator

6.1.1 Use case diagram

As the simulator will runs automatically, it will not have a big set of functionalities. That's why we just have one use case.

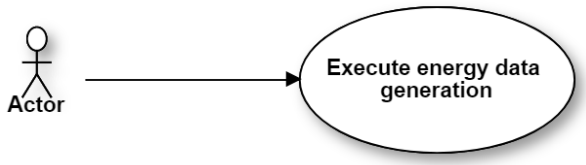


Figure 1: Simulator use case

6.1.2 Use case specification

In order to specify the use cases, I will define them using the following structure:

- Use case #[use case number]: [use case title]
- Principal actor: [principal actor of the use case]
- Goal: [the reason why this requirement is necessary]
- Summary: [use case description]
- Typical course of events: [typical interactions sequence]

Use case #1	Energy data generation
Principal actor	Everyone.
Goal	Simulate the trains consumed energy.
Summary	The user runs the simulator that will generate the energy consumption of the specified trains.
Typical course of events	1. The user runs the script that will execute the energy consumption simulation.

6.1.3 Requirements

Functional requirements

In order to explain the requirements, I will introduce this information for each functionality that the system has to realize:

- Requirement #[requirement number]: [requirement title]
- Description: [requirement description]
- Justification: [the reason why this requirement is necessary]

Requirement #1	Energy consumption data generation
Description	The users want to generate the trains energy consumption.
Justification	It is necessary to have consumed energy data from the trains in order to fill the database with information.

Non-functional requirements

I will use the same structure to present, so as to present the non-functional requirements.

- Requirement #[requirement number]: [requirement title]
- Description: [requirement description]
- Justification: [the reason why this requirement is necessary]

Requirement #1	Coherence energy data
Description	The simulator has to provide a simulated energy consumption data that have a coherence with the real ones
Justification	It is necessary to maintain the coherence between the applications that the clients have at this moment and the one that we have developed

6.1.4 Class diagram

We have divided the Simulator in four big modules, as we can see at the following class diagram.

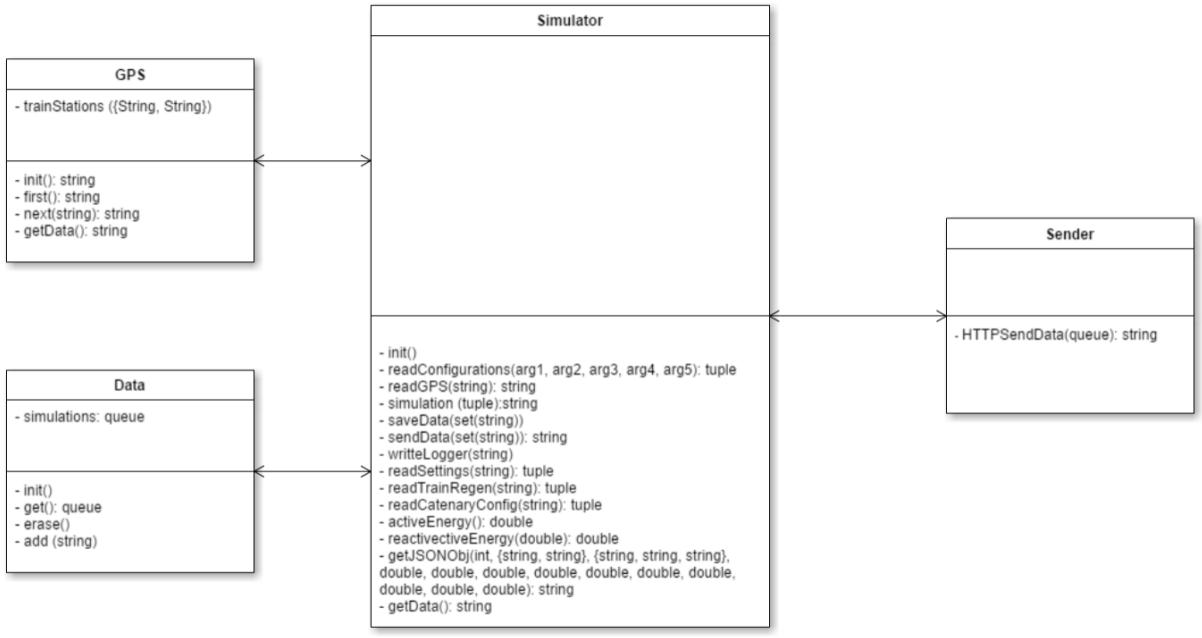


Figure 2: Simulator class diagram

6.2 Data manage modules

I have considered to group both databases, the APIs and the Client because they have the Front End connection. The Client will make the union between them, so we can suppose that they will share the requirements.

6.2.1 Use cases diagrams

In this section we can see all the interactions that the user can do with the application. In order to explain them, I have done four different diagrams, divided by user roles, that will show the functionalities that every type of user can do.

First, we will start with an anonymous user, who is not registered to the application.

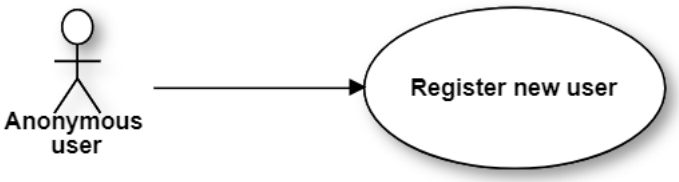


Figure 3: Anonymous user use cases diagram

As we can see in the Figure 3, the anonymous user can just register himself to the system.

Secondly, we have the system administrator. This user has the responsibility to manage all the companies that the system have and their users. It needs this control due to the fact that every user registered is defined as not authorized and the lower role that the system can has per default and it is the only one who can add new companies to the system.

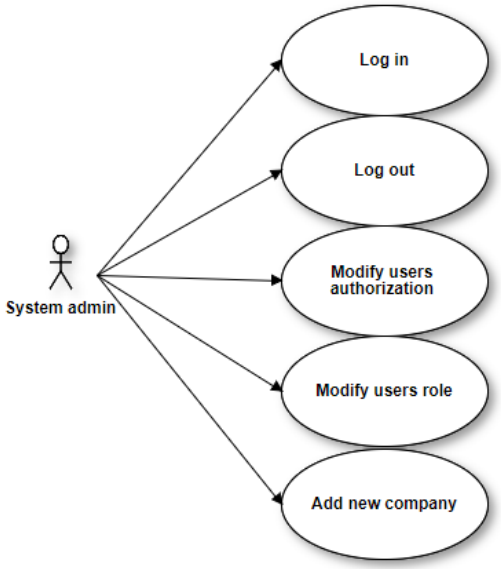


Figure 4: system administrator use cases diagram

Thirdly, we have the company admin use case. As we see in the Figure 5, it has almost all the functionalities that the system provides to us.

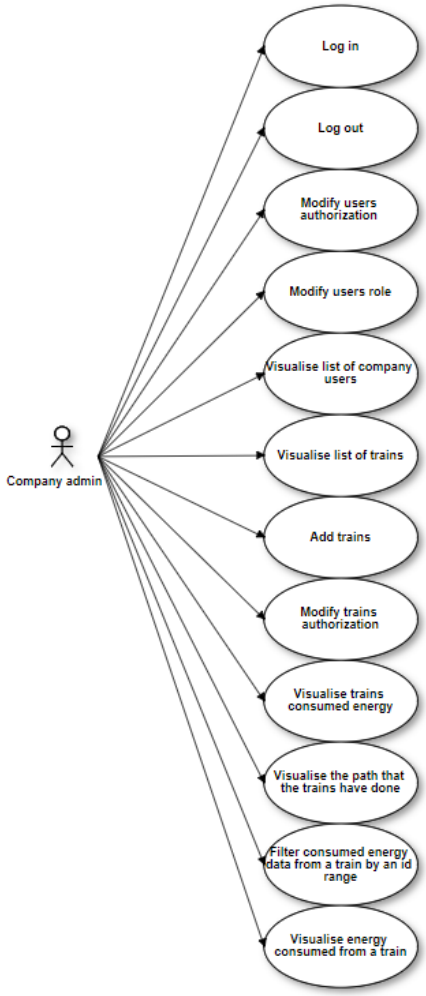


Figure 5: company administrator use cases diagram

Finally, we have the use cases diagram for the company user. As we can see in the Figure 6, we can see that this user role has the majority of the functionalities that the company administrator have, this is due to the fact that the company administrator is a user company with a higher rank, and consequently, it has more responsibilities and powers.

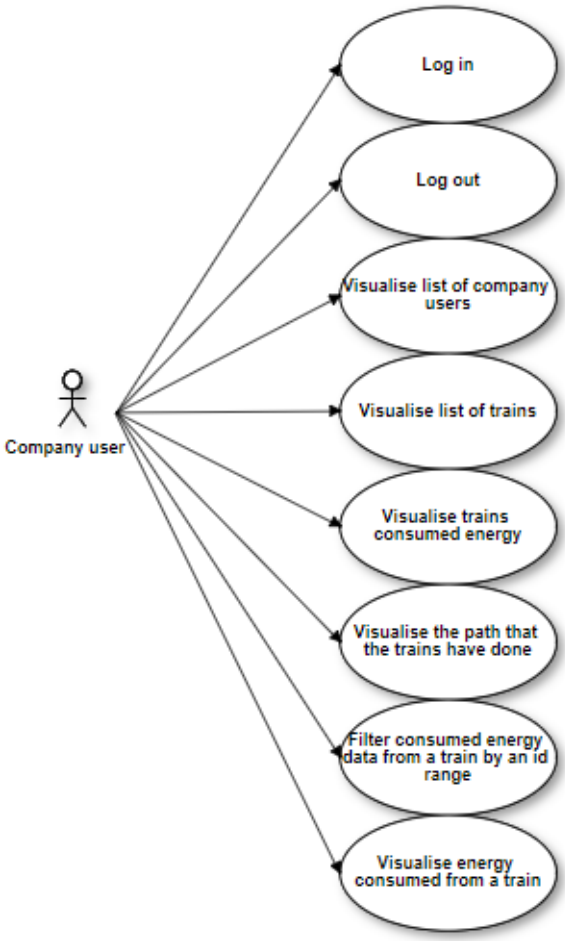


Figure 6: company user use cases diagram

6.2.2 Use cases specifications

In order to specify the use cases, I will define them using the following structure:

- Use case #[use case number]: [use case title]
- Principal actor: [principal actor of the use case]
- Goal: [the reason why this requirement is necessary]
- Summary: [use case description]
- Typical course of events: [typical interactions sequence]

Use case #1	Register new user
Principal actor	Everyone.
Goal	Simulate the trains consumed energy.
Summary	The user runs the simulator that will generate the energy consumption of the specified trains.
Typical course of events	1. The user runs the script that will execute the energy consumption simulation. 2. The system confirms the transaction.

Use case #2	Log in
Principal actor	Everyone.
Goal	Access to your system account.
Summary	The user access to the web service.
Typical course of events	1. The user request to access to the application with his credentials. 2. The system returns the access token.

Use case #3	Log out
Principal actor	Everyone.
Goal	Log out from your system account.
Summary	The user log out from the web service.
Typical course of events	1. The user has logged in to the system. 2. The user request to log out from the application. 3. The system delete the access token. 4. The system communicates that the user has logged out.

Use case #4	Modify users' authorisation
Principal actor	System admin and company admin.
Goal	Authorise the access to the system to the users.
Summary	The administrators can modify the access to the system to other users.
Typical course of events	1. The administrator has correctly logged in. 2. The system shows the list of users. 3. The administrator select enable/disable the user authorisation. 4. The system communicates if the transaction has been correctly done.

Use case #5	Modify users' roles
Principal actor	System admin and company admin.
Goal	Modify the roles of the users. They can be set as a company user or company admin.
Summary	The users can have different roles, so we need a function that allow to the administrators to control it.
Typical course of events	1. The administrator has correctly logged in. 2. The system shows the list of users. 3. The administrator select enable/disable the role company admin from one user. 4. The system communicates if the transaction has been correctly done.

Use case #6	Add a new company
Principal actor	System admin.
Goal	Add new companies to the system.
Summary	New companies have to be added to the system.

Use case #6	Add a new company
Typical course of events	1. The system admin has correctly logged in. 2. The administrator select register new company. 3. The system shows the company's registration screen. 4. The system admin register the new company. 5. The system confirms the registration.

Use case #7	Show company users
Principal actor	Company admin and company user.
Goal	Show the users from the same company.
Summary	The users can see the users from the company that they work for.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users.

Use case #8	Show company trains
Principal actor	Company admin and company user.
Goal	Show the trains from company.
Summary	The users can see the trains from the company that they work for.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company.

Use case #9	Add new company trains
Principal actor	Company admin.
Goal	Add new trains to the company.
Summary	The company admin can add new trains to the company that he works for.
Typical course of events	1. The administrator has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company. 5. The administrator writes the new id of the train and send the register request. 6. The system notify the correct registration of the train.

Use case #10	Modify trains' authorisation
Principal actor	Company admin.
Goal	Authorise the train run and do trips throw the rails network.
Summary	The users can authorise the trains from his company to run and do routes throw the rails network.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company. 5. The administrator select enable/disable the train authorisation. 6. The system communicates if the transaction has been correctly done.

Use case #11	Visualise the train's route
Principal actor	Company admin and company user.
Goal	Show the trains' route.
Summary	The users can see the route that a train has done.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company. 5. The user select the id of a train. 6. The system shows the route of the train.

Use case #12	Show consumed energy from a train
Principal actor	Company admin and company user.
Goal	Show the consumed energy from a train.
Summary	The users can list the energy consumed by a train.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company. 5. The user select the id of a train. 6. The system shows the energy consumed by the selected train.

Use case #13	Filter consumed energy from a train
Principal actor	Company admin and company user.
Goal	Filter the energy data.

Use case #13	Filter consumed energy from a train
Summary	The users can filter the energy data by id.
Typical course of events	1. The user has correctly logged in. 2. The system shows the list of users. 3. The user select the trains management option. 4. The system shows the trains of the company. 5. The user select the id of a train. 6. The system shows the train's energy data. 7. The user specifies the range of ids. 8. The system shows the train's energy data content at the range.

6.2.3 Requirements

Functional requirements

In order to explain the requirements, I will introduce this information for each functionality that the system has to realize:

- Requirement #[requirement number]: [requirement title]
- Description: [requirement description]
- Justification: [the reason why this requirement is necessary]

Requirement #1	Register new user
Description	The users want to register themselves in the system.
Justification	It is necessary to have access to the system.

Requirement #2	Add new company
Description	The users want to add new companies to the system.
Justification	It is necessary to have companies in order to add new users.

Requirement #3	Add new train
Description	The users want to add new trains to the company that they work for.
Justification	It is necessary to have created the trains before we execute the simulator. We can not have energy data from trains who does not exist.

Requirement #4	Add new energy data
Description	The trains (IoT) want to add new energy data to the system.
Justification	It is necessary due to the fact that the users want to see the energy consumed by their companies, and this is the only way to add new data to the databases.

Requirement #5	Log in
Description	The users want to log in to the system.
Justification	It is necessary access to the system.

Requirement #6	Log out
Description	The users want to log out to the system.
Justification	It is necessary to log out from the system.

Requirement #7	Modify user access authorization
Description	The company administrators want to modify the access authorization of the users.
Justification	To manage the access from the company employees to the system.

Requirement #8	Modify user role
Description	The company administrators want to modify the user's role.
Justification	It is necessary to modify the powers that the users have throw the system.

Requirement #9	Show train path
Description	The users want to see the path that the train has followed.
Justification	It is necessary to show the train stations that the user has been.

Requirement #10	Show train energy data
Description	The users want to see the trains energy data.
Justification	It is necessary to check the amount of energy that they have consumed.

Requirement #11	Show companies
Description	The users want to see the different companies that the system have.
Justification	It is necessary to select your company in order to log in

Requirement #12	Show users
Description	The users want to see the users from the same company.
Justification	It is necessary to see who works at your company.

Requirement #13	Show Blockchain data
Description	The users want to see the data saved at the Blockchain.
Justification	It is necessary to see the data who is saved there. It will be the one who will be completely secure and transparent.

Requirement #14	Show range of energy data
Description	The users want to see a range of energy data.
Justification	When a train has a big amount of energy data documents stored, the user will see just the last 30, so this feature will allow to him to see another range.

Non-functional requirements

I will use the same structure to present, so as to present the non-functional requirements.

- Requirement #[requirement number]: [requirement title]
- Description: [requirement description]
- Justification: [the reason why this requirement is necessary]

Usability requirements

Requirement #1	Interface comprehensible
Description	The interface is easy to use, to understand and it contains the minimum functionalities, so as to simplify the use.
Justification	Due to the fact that the program will be used frequently,

Requirement #2	Easy to see consumed energy data
Description	It has to be easy to see the energy data consumed by a train.
Justification	The user has to be able to see the consumed energy by a train without any problems, even if it is the first time that he/she uses the application for the first time.

Requirement #3	Formal language
Description	The application has to be written in a formal and accurate language.
Justification	So as to increment the usability and the comprehension of the functionalities of the application.

Requirement #4	Browser adaptability
Description	The application will work in every browser.
Justification	As it is a web application, we do not know where the user will execute it, so it is important to ensure the correct execution at every browser.

Security and legal requirements

Requirement #5	Law accomplishment
Description	The application will accomplish the French data protection law.
Justification	The application will have to accomplish with the France laws, so as to be legal.

Requirement #6	Access restricted
Description	The users will just access to the functionalities that their role allows them to use.
Justification	The user actions have to be controlled by permissions, so as to avoid wrong uses.

Availability and performance requirements

Requirement #7	Fluent interface
Description	The interface has to be fluent, it should not have interruptions while the user uses it.
Justification	The application will be used frequently, so it should be fast.

Requirement #8	Failure security
Description	A system failure does not have to affect seriously the service provided to the users.
Justification	A bad request or a database error should not affect to the application performance.

Scalability requirements

Requirement #9	More organisations scalability
Description	The system has to be fluent, although there are a big amount of companies at the system.
Justification	In the future, more companies will be added at the system, so it should ensure that it will still work fluently.

Requirement #10	Project scalability
Description	The project has to be flexible and adaptable to new modules, so as to add new functionalities or modify the one that already exists.
Justification	It will be necessary to add new features.

7. PROJECT DESIGN

7.1 Architecture

At this subsection I am going to present the actual architecture of the SNCF system and then, the one that we have done to solve the problem and the incoming new requirements, so as to present you a global vision of the project.

7.1.1 Initial Architecture

In this section, I will introduce the actual SNCF data management system architecture where, as we can see in the Figure 7, is composed by three different parts, the IoT, SNCF server and the ERESS[15].

Firstly we have the SNCF trains, the actors who compose the Internet of Things network. These trains will consume energy, while they run through the French rail network, and they will send it to the SNCF Servers via GSM, the second part of the actual architecture. The SNCF servers are provided with a centralized database, that will store all the energy data, and an API who allows to manage it. Finally, this amount of data will be shared with the ERESS organization, a non-profit organization, jointly owned by its partners, committed to the development, implementation and supply of the energy settlement. This will be the organization who will invoice the energy consumed by SNCF using the servers' saved data.

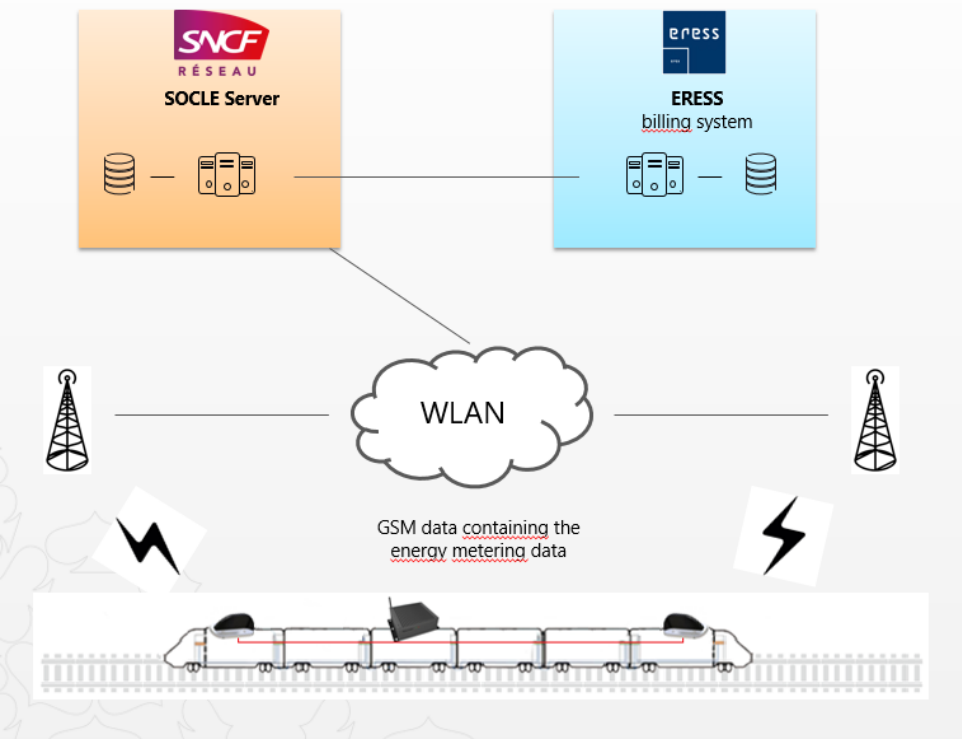


Figure 7: actual SNCF data management system architecture

7.1.2 Final Architecture

After presenting the initial architecture, I will introduce yourselves to the SNCF energy data management system architecture that we have built in order to adapt the actual one to the SNCF incoming changes. So, we have passed from three to five different parts, the IoT, the Socle server, the NoSQL database, the Blockchain database and the ERESS organization.

Firstly, we need the trains consumed energy data, but, as we can not access to this information, the first part of our presented architecture is the simulator who will generate periodically the energy consumption of the trains.

Secondly, there is the Socle server who communicates with the AMQP broker, technology that will be explained at the technologies section. This module, allows us to send the energy data in an efficient way to both databases.

Thirdly, we have the IBM Cloudant, a NoSQL database who will also be explained at the technologies section. This module consists in a centralized database and the API who manages it from SNCF companyI. This part will store all the information of the system, including the sensitive information, that its own correctness and trustworthiness will be supported by the one that will be stored at the Blockchain.

Fourthly, there is the Blockchain module, a decentralized database that will store all the sensitive information. This part, as we can appreciate at the Figure 8, involves all the companies from the system because each of them have a database mirror and their own API to manage the information that the chain stores.

Finally, the ERESS organization will invoice the companies using the Blockchain's saved data and the API who allows them to access to all the French rails network consumed energy.

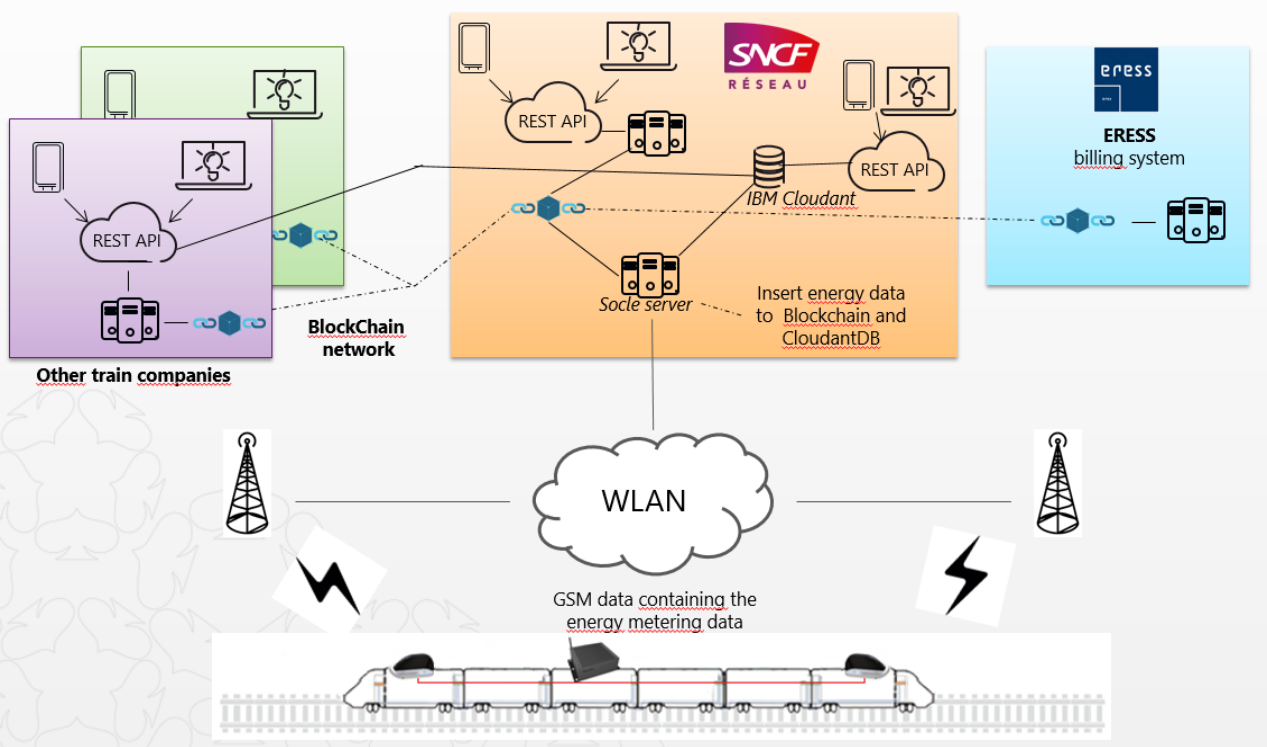


Figure 8: final SNCF data management system architecture

7.2 NoSQL database design

After having in mind our proposed architecture, I would like to introduce the NoSQL database design. It is simple, as it just saves directly JSON documents, we do not have to be worried with the foreign keys, restrictions… We will manage everything at the APIs. That is why we just have four different types of documents, user, company, train and cebd (energy data document). At this point, the only thing to contrast is the _id and _rev attribute, where the pair is unique and both of them are automatically generated by IBM Cloudant.

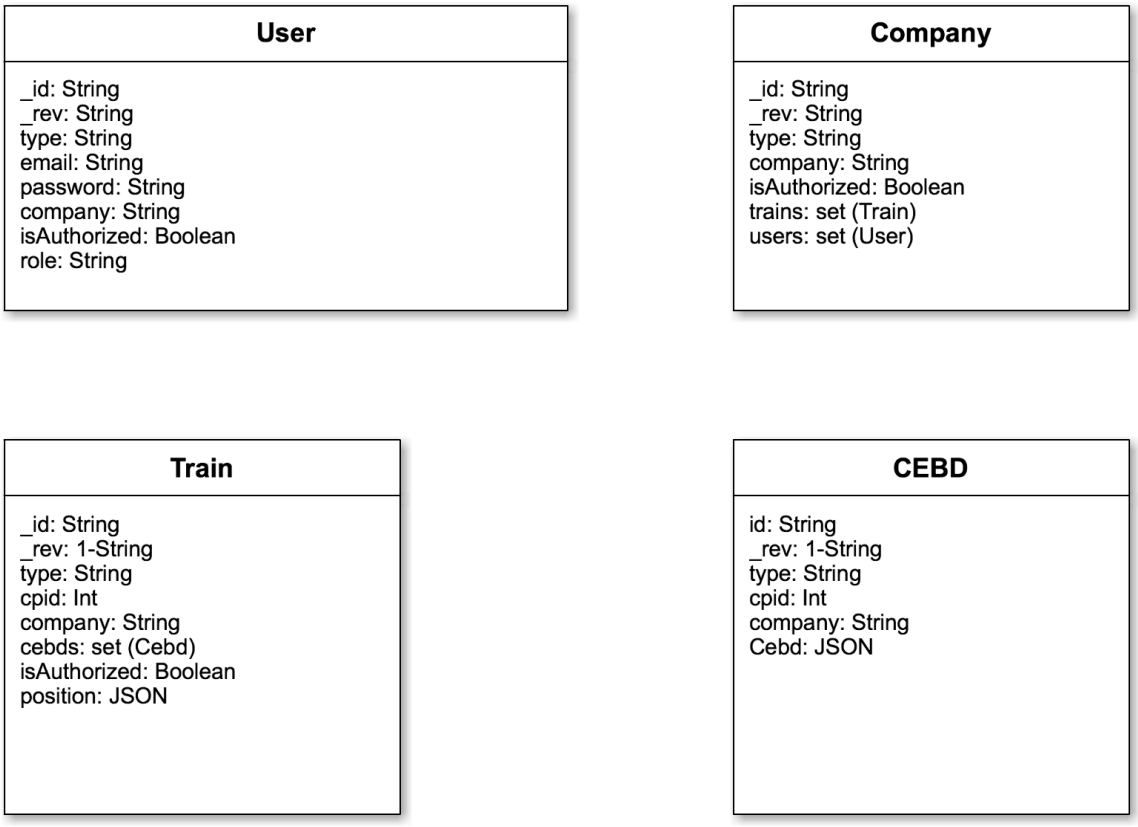


Figure 9: IBM Cloudant database design

7.3 Technologies

As it is done in every software project, we have thought in some different technologies that could fit to the system requirements, apart of the ones that we have said in the previous section to use.

In order to introduce yourselves to the used technologies, I have divided this section in three big groups sorted by the different modules of the project, Simulator, data storage and Client. Where I will introduce the different technologies that we had contemplated to use.

7.3.1 Simulator

At this part of the project, we had to bear in mind that we would have to deal with the Internet of Things (IoT), due to the fact that the trains provide the energy data. So, as we will simulate this part using a Simulator, we can conclude that this module will be the IoT part of the project. So firstly, I will introduce yourselves to the Internet of Things.

Internet of Things

Internet of things [16] is the inter-networking of physical devices that have electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data between them.

The IoT is significant because an object that can represent itself digitally becomes something greater than the object by itself. No longer does the object relate just to its user, but is now connected to surrounding objects and database data. Finally, when many objects act in unison, they are known as having ambient intelligence.

In order to develop a software who will generate the trains' energy data we have thought to use Java or Python. But, due to the fact that Java has to be compiled before you can execute the code, we concluded to use Python as a code language. As a consequence of this decision, we have used the Pytest framework, so as to test every piece of code that we have written, and Jenkins to execute automatically the Pytest tests built.

Python 3.6

Python [17] is a widely used high-level programming language for general-purpose programming. As it is an interpreted language, Python has a design philosophy that emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

Pytest

Pytest [18] is a mature full-featured Python testing tool that helps you to write and check your code. This framework makes it easy to write small tests, as well as scales to support complex functional testing for applications and libraries.

We chose this testing tool because we had colleagues with experience in that framework who could help us if we needed.

Jenkins

Jenkins [19] is an open source automation server written in Java, who helps to automate the non-human part of software development process, with continuous integration and facilitating technical aspects of continuous delivery.

Finally, at this module we have used one more technology which will help us to send the energy data to the other network actors. This tool will send the information without useless delays using the Advanced Message Queuing Protocol (AMQP) who will allow us to have one queue for each database. I will explain the reason to use it in the scope section.

RabbitMQ

RabbitMQ [20] is an open source message broker software that implements the Advanced Message Queuing Protocol. The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Almost all the major programming languages have Client libraries to interface with the broker.

7.3.2 Data stockage

After doing a detailed analysis of the problem, we figure out that we have to implement a solution who gives a full transparency and confidence of the data. This, is due to the fact that this system will involve a big amount of consumed energy, and consequently, a lot of money of different companies.

So, bearing in mind these requirements and considering the state of art conclusions, we have realized that could be a good idea to use the emerging Blockchain technology, as this will help us to achieve some of the needs that our system should cover.

This will not be as easy as we think, this type of Database has the problem that is really heavy, so it will not be efficient to save all the energy data in it. That is why we will have to add another database to store all the information that will not be saved at the Blockchain. So, we will store at the chain the sensitive information, in other words, the train energy consumed, and at the other database, the rest of energetic information, like the employees, speed of the train, etc.

In conclusion, we will have to choose two different database implementations, one for the general data and another one for the sensitive one, that will be based on the Blockchain technology.

7.3.2.1 General data database

In this part, we have analysed the typical and mature databases. The ones we have considered that could fit to our requirements, that can be easily deployed at internet and who are efficient JSON type documents.

SQL database

Our first option was to use a relational database based on Structured Query Language (SQL), the standard language for storing, manipulating and retrieving data in databases.

This would be a collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.

The advantages that this technology would bring to us is the simplicity, flexibility and a structured database.

The main disadvantage is that can be slow. If there are many tables utilizing relationships, the responsiveness of data queries can be delayed. In addition, relational databases scale up well, but do not scale out well, making storage expensive.

NoSQL database

The other option that we thought that could fit was to use a NoSQL database.

NoSQL describe an approach to database design that implements a key-value store, document store, column store or graph format for data. NoSQL contrasts to databases that adhere to SQL's relational methods, where data are placed on tables and data schema are carefully designed before the database is built.

The advantages that this technology would bring to us is that NoSQL databases are generally more scalable than relational ones and performance is generally not an issue. They are designed to expand transparently and horizontally using low-cost hardware.

The main disadvantage is that they cannot handle the analytical processing of the data or joins, which are common requirements for the Internet of Things applications. They employ low-level query languages, and do not accommodate transactions where data integrity needs to be preserved.

As we want a database who scales horizontally, we have found out that the NoSQL databases would fit better to our project. So, at this point we started to consider some different NoSQL databases, and we found two interesting ones. MongoDB [21] and IBM Cloudant [22] databases, where I define at the following sections.

MongoDB

MongoDB is an open source document-oriented database, classified as a NoSQL database.

Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and JSON-like documents. Where the documents comprise sets of key-value pairs and are the basic unit of data. Then, collections contain sets of documents and function as the equivalent of relational database tables.

IBM Cloudant

Cloudant is an IBM software product, which is primarily delivered as a cloud-based service. Cloudant is an open source NoSQL database based on the Apache-backed CouchDB and the open source BigCouch projects.

The differential part of Cloudant is that it provides a data management, search, and analytics engine designed for web applications. So, we have that it extends CouchDB in several ways. Firstly, it provides a chained MapReduce Views. Secondly, it allows the usage of Java for CouchDB map reduce analytics. And finally, it provides application programming interface keys for programmatic access to CouchDB database.

As CloudantDB involves some different technologies, I consider convenient to introduce them to yourselves.

CouchDB

CouchDB [23] is a “NoSQL” database which does not have schemas, or rigid pre-defined data structures such as tables. The data is stored as a JSON document and his structure can change dynamically to accommodate the evolving needs.

BigCouch

BigCouch [24] is an open source, highly available, fault-tolerant, clustered and API-compliant version of Apache CouchDB. Which allows the users to create clusters of CouchDBs that are distributed over an arbitrary number of servers, while it appears to the end-user as one CouchDB instance, it is in fact one or more nodes in an elastic cluster, acting in concert to store and retrieve documents, index and serve views, and serve CouchApps.

MapReduce

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

A MapReduce program is composed of a Map method, that performs filtering and sorting, and a Reduce method that performs a summary operation. So, the MapReduce System organizes the processes by marshaling the distributed servers, running the tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

So, before we chose one of them, we should see the differences and analyze them in order to know which one would fit better. As at first sight, both of them could be good.

MongoDB vs CloudantDB

Cloudant is a document based NoSQL database, just like MongoDB. It is a wrapper around CouchDB with more functionalities. Like CouchDB, it is a RESTful service with a web-based GUI that is capable to do all your basic CRUD operations and which it is designed around map/reduce functions.

The documents are not organized in collections like in Mongo. Instead, every database, simply stores all the documents together, so if you want to query only a certain subset, then you need to create a field within each document that will act to distinguish it from other types of documents. You can do it defining MapReduce views, pre-defined queries, to filter your data by given fields.

So, we have that Cloudant is a distributed CouchDB that has several features that set it apart. It allows you to use full text search indexes based on Apache Lucene search, Cloudant Geospatial for dedicated deployments and Cloudant Query.

Cloudant Query is an additional way in which Cloudant allows you to get at your data. It is Cloudant's attempt at a declarative query language based on MongoDB's that is more intuitive for those from a SQL or MongoDB background. This is slightly faster to build indexes than using the traditional MapReduce views so it is the recommended path provided.

MongoDB also has capped collections and time to live collections while Cloudant does not. However, you could certainly replicate the behavior of these collections without too much effort in your code if you so desired.

So, as we have seen, IBM Cloudnat brings us many more functionalities than MongoDB, so we have decided to use the IBM option.

Due to the fact that we will use IBM Cloudant, I have considered important to present the IBM tool who will let us to deploy this database on the internet, the IBM Bluemix, and the NodeJS, the programming language that we will use to code the database and the API who will allow for the users to manage the saved data.

IBM Bluemix

IBM Bluemix [25] is a cloud platform to help developers build and run modern apps and services. It provides developers with instant access to the compute and services they need to launch quickly, iterate continuously and scale with success.

NodeJS

Node.js [26] is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side. So, Node.js enables JavaScript to be used for server-side scripting, and runs scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js has become one of the foundational elements of the "JavaScript everywhere" paradigm [27], allowing web application development to unify around a single programming language, rather than rely on a different language for writing server side scripts.

Finally, we have built a testing module who will verify the correctness of our code, just like the Simulator. Nevertheless, at this case we have used two different technologies, Mocha framework and the Chai assertion library. Both of them use NodeJS as programming language.

Mocha

Mocha [28] is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases which is hosted on GitHub.

Chai

Chai [29] is a Behavior-Driven Development and Test-Driven Development assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

7.3.2.2 Blockchain

Blockchain is a distributed database that is used to maintain a continuously growing list of records, called blocks. Where each block contains a timestamp and a link to a previous block. This type of databases are typically managed by a peer-to-peer network collectively adhering to a protocol for validating new blocks.

By design, Blockchain are inherently resistant to modification of the data. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks and a collusion of the network majority. Functionally, a Blockchain can serve as an open, distributed ledger

that can record transactions between two parties efficiently and in a verifiable and permanent way. Moreover, the ledger itself can also be programmed to trigger transactions automatically.

The different Blockchain implementations are divided in two big groups, the public and private Blockchains. Where the sole distinction between them is related to who is allowed to participate in the network, execute the consensus protocol and maintain the shared ledger.

In one hand, a public Blockchain network is completely open and anyone can join and participate in the network. The network typically has an incentivising mechanism to encourage more participants to join the network.

On the other hand, a private Blockchain network requires an invitation and must be validated by either the network starter or by a set of rules put in place by the network starter. Businesses who set up a private Blockchain, will generally set up a permissioned network who will places restrictions on who is allowed to participate in the network in certain transactions. Participants need to obtain an invitation or permission to join. This access control mechanism could vary: existing participants could decide future entrants; a regulatory authority could issue licenses for participation; or a consortium could make the decisions instead. Once an entity has joined the network, it will play a role in maintaining the Blockchain in a decentralised manner.

As it exists some public and private Blockchain implementations and some that could fit the two types, we will analyze the most important ones, and find out the one that will fit better for us.

Public Blockchains

Bitcoin

Bitcoin [30] is a cryptocurrency and a digital payment system who is based on Blockchain technology, and at the moment, is the most important public Blockchain implementation.

The system is peer-to-peer, and transactions take place between users directly, without an intermediary. These transactions are verified by network nodes and recorded in a public distributed ledger, the Blockchain. Since the system works without a central repository or single administrator, Bitcoin is called the first decentralized digital currency.

Private Blockchains

Corda

Corda [31] is an open-source distributed ledger, which supports smart contracts, and it's especially geared towards the financial world as it handles more complex transactions and restricts access to transaction data. Although it is inspired by Blockchain databases, and is expected to have many of the benefits of Blockchains, it is not a Blockchain.

The aim of Corda is to provide a platform with common services to ensure that any services built on top are compatible between the network participants.

Hyperledger

Hyperledger [32] is an open source collaborative effort created to advance cross-industry Blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing and Technology.

There are five Hyperledger Frameworks to choose, in order to choose one, I will write a short description of each one.

Hyperledger Sawtooth

Hyperledger Sawtooth is a modular platform for building, deploying, and running distributed ledgers. It includes a novel consensus algorithm, Proof of Elapsed Time (PoET), which targets large distributed validator populations with minimal resource consumption.

Hyperledger Iroha

Hyperledger Iroha is a business blockchain framework designed to be simple and easy to incorporate into infrastructural projects requiring distributed ledger technology.

Hyperledger Fabric

Intended as a foundation for developing applications or solutions with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play.

Hyperledger Burrow

Hyperledger Burrow is a permissionable smart contract machine. It provides a modular blockchain client with a permissioned smart contract interpreter built in part to the specification of the Ethereum Virtual Machine (EVM).

Hyperledger Indy

Hyperledger Indy provides tools, libraries, and reusable components for providing digital identities rooted on blockchains or other distributed ledgers so that they are interoperable across administrative domains, applications, and any other silo.

After defining this few Hyperledger Frameworks, we see that the one who would better fit for our project is the Hyperledger Fabric, as it gives us a completely flexibility of our project.

Private and private Blockchains

Ethereum

Ethereum [33] is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. No one controls or owns Ethereum, and unlike the Bitcoin protocol, Ethereum was designed to be adaptable and flexible. It is easy to create new applications on the Ethereum platform, and with the Homestead release, it is now safe for anyone to use those applications.

Conclusion

To conclude, we see that we have three different Blockchain technologies to choose, Bitcoin, Corda, Ethereum and finally, Hyperledger Fabric.

Considering that our project involves not just economic transactions and that we want a private Blockchain, we can see that Bitcoin and Corda would not fit to our requirements.

Then, between the Ethereum and Hyperledger, we have that the most fundamental difference is the way they are designed and their target audience.

Ethereum with its Ethereum Virtual Machine, smart contract and public Blockchain is mostly targeted towards applications that are distributed in nature and are for mass consumption.

On the other hand, Hyperledger Fabric has a very modular architecture and provides a lot of flexibility in terms of what you want to use and what you don't. It's pretty much ala carte and is targeted at businesses wanting to streamline their process by leveraging blockchain technology.

So, finally, we have chosen the Hyperledger Fabric Framework to implement our Blockchain, due to the fact that it gives us more flexibility and adaptability to our requirements.

Hyperledger Fabric v1.0

In order to develop using the last release of the Hyperledger Fabric we have used the 1.0 version. Which has been developed while we were coding our project. Due to this fact, we had the opportunity to help the community to improve this Blockchain implementation.

The Hyperledger Fabric uses two different types of programming language, the JavaScript and Go, which I will define at the following lines.

JavaScript

JavaScript [34] is a high-level, dynamic, weakly typed, object-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make web pages interactive and provide online programs, including video games. The majority of web sites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine.

Go

Go [35] is a free and open source programming language created by Google in 2007. It is a compiled, statically typed language in the tradition of Algol and C, with garbage collection, limited structural typing, memory safety features and Communicating sequential processes style concurrent programming features added.

Basic components of an Hyperledger Fabric network

Block

An ordered set of transactions that is cryptographically linked to the preceding block(s) on a channel.

Chain

Is a continuously growing list blocks.

Chaincode

Often called Smart Contract, is software, running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets.

Channel

A channel is a private Blockchain overlay which allows for data isolation and confidentiality. A channel-specific ledger is shared across the peers in the channel, and transacting parties must be properly authenticated to a channel in order to interact with it.

Consensus

A broader term overarching the entire transactional flow, which serves to generate an agreement on the order and to confirm the correctness of the set of transactions constituting a block.

Hyperledger Fabric CA

Hyperledger Fabric CA is the default Certificate Authority component, which issues PKI-based certificates to network member organizations and their users. The CA issues one root certificate to each member and one enrollment certificate to each authorized user.

Genesis Block

The configuration block that initializes a Blockchain network or channel, and also serves as the first block on a chain.

Member

A legally separate entity that owns a unique root certificate for the network. Network components such as peer nodes and application clients will be linked to a member.

Membership Service Provider

The Membership Service Provider (MSP) refers to an abstract component of the system that provides credentials to clients, and peers for them to participate in a Hyperledger Fabric network. Clients use these credentials to authenticate their transactions, and peers use these credentials to authenticate transaction processing results (endorsements). While strongly connected to the transaction processing components of the systems, this interface aims to have membership services components defined, in such a way that alternate implementations of this can be smoothly plugged in without modifying the core of transaction processing components of the system.

Membership Services

Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network. The membership services code that runs in peers and orderers both authenticates and authorizes blockchain operations. It is a PKI-based implementation of the Membership Services Provider (MSP) abstraction.

Ordering Service

A defined collective of nodes that order transactions into a block. The ordering service exists independent of the peer processes and orders transactions on a first-come-first-serve basis for all channel's on the network. The ordering service is designed to support pluggable implementations beyond the out-of-the-box SOLO and Kafka varieties. The ordering service is a common binding for the overall network; it contains the cryptographic identity material tied to each Member.

Peer

A network entity that maintains a ledger and runs chaincode containers in order to perform read/write operations to the ledger. Peers are owned and maintained by members.

7.3.3 Client

Finally, we have the Front end module. The one who will allow to the users to manage all the information that the system contains. At this project we have decided to use Angular 2. We choose that technology because we had experience with previous version and due to the fact that the new release has better performance and it is more efficient.

Angular 2

Angular [36] is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations to address all of the parts of the developer's workflow while building complex web applications. Angular is a complete rewrite from the same team that built AngularJS.

TypeScript

TypeScript [37] is a free and open-source programming language developed and maintained by Microsoft. TypeScript may be used to develop JavaScript applications for client-side or server-side (Node.js) execution.

TypeScript is designed for development of large applications and compiles to JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.

8. IMPLEMENTATION

At this section, is where the defined specification and design requirements are applied in order to achieve the main goal of the project, to show the consumed energy by the trains. At the following paragraphs I will explain the implementation process as well as the result.

Show trains' consumed energy

Everything starts with a simulator who will generate energy consumption data from a specified train who does a route throw France. As I have explained at the technologies section, it is implemented using Python and it will generate a JSON document who will contain the consumed energy from a train, in a specific period of time (Figure 10).


```
{"Cebd": {"VoltageMax": 58703326560, "Measures": {"1": {"Quality": 1, "Voltage": 27566387082.0,
"CurrentMax": 1630647960, "emf": {"SerialNumber": 123456, "IsAlive": true, "EmfId": 1,
"CatenaryType": 1}, "CurrentMin": -99182011, "VoltageMax": 58703326560, "ConsumedActiveEnergy":
4053763.44, "Current": 765732974.5, "RegeneratedReactiveEnergy": 9.59741598150279e-08,
"RegeneratedActiveEnergy": 4053763.44, "ConsumedReactiveEnergy": 9.59741598150279e-08,
"VoltageMin": -3570552396}, "2": {"Quality": 1, "Voltage": 1669635108.0, "CurrentMax":
1248092162, "emf": {"SerialNumber": 123457, "IsAlive": true, "EmfId": 1, "CatenaryType": 2},
"CurrentMin": -1155334656, "VoltageMax": 44931317832, "ConsumedActiveEnergy": 1576463.56,
"Current": 46378753.0, "RegeneratedReactiveEnergy": 3.732328437251086e-08,
"RegeneratedActiveEnergy": 1576463.56, "ConsumedReactiveEnergy": 3.732328437251086e-08,
"VoltageMin": -41592047616}}, "CebdId": 11, "Voltage": 8555639472.0, "RegeneratedActiveEnergy":
5630227.0, "CurrentMax": 1630647960, "RegeneratedReactiveEnergy": 1.3329744418753876e-07,
"VoltageMin": -41592047616, "End": {"Satellite": 0, "Direction": 0.0, "Longitude": 14.126161,
"GpsStatus": 0, "Latitude": 40.827566, "Rail_Network": "Trenitalia", "Date":
"2017-10-12T12:44:07.061782+00:00", "Altitude": 0.0, "Speed": 9.09134606235722}, "Current":
765732974.5, "ConsumedReactiveEnergy": 1.3329744418753876e-07, "Quality": 1, "Begin":
{"Satellite": 0, "Direction": 0.0, "Longitude": 11.781181, "GpsStatus": 0, "Latitude": 47.94581,
"Rail_Network": "DeutscheBahn", "Date": "2017-10-12T12:44:05.061782+00:00", "Altitude": 0.0,
"Speed": 5.366117005199467}, "CurrentMin": -99182011, "ConsumedActiveEnergy": 5630227.0},
"Header": {"TrpPeriod": 2, "RecordTime": "68360273848048367", "Crc": 286920174, "Cpid": 301}}
```


Figure 10: consumed energy document from 301 generated by the simulator

So, every time that we execute the simulator, it will send, periodically, this type of documents (Figure 10) to the RabbitMQ queues, who will automatically send the information to the Cloudant IBM and Hyperledger Fabric databases.

Finally, if everything has gone well, we can access to the web service, using our account, and check the amount of energy that the trains of our company has consumed (Figure 11).

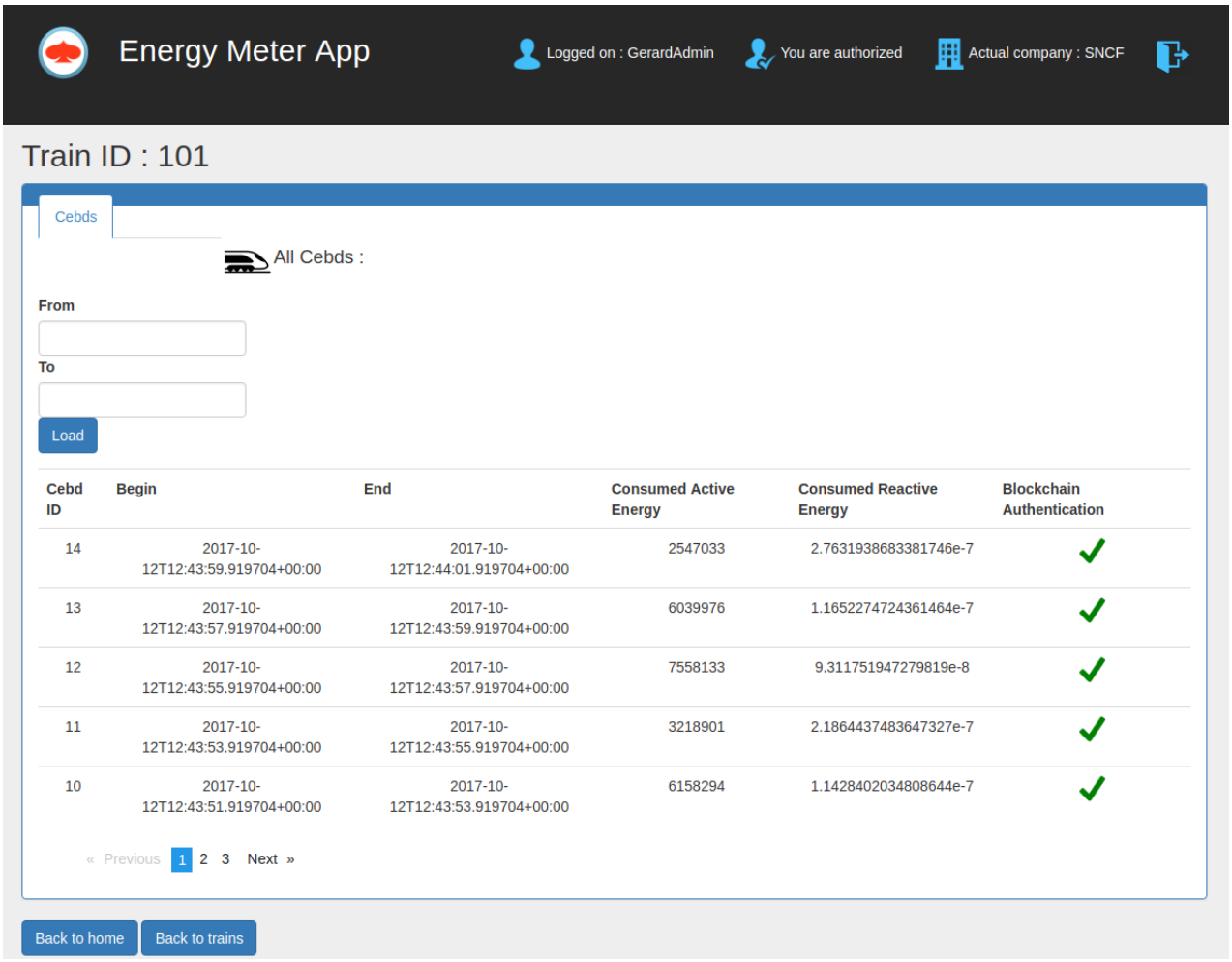


Figure 11: energy consumption screen from the train 101

9. TESTING

The testing part is really important process at the software development, that's why I considered important to sum up the technologies and methodologies that we have followed.

Here I present the framework used for each module to test the correct execution of the code.

Simulator

As I have explained at the technology section, we have developed the Simulator using the Python 3 release. So, searching a bit, we found a framework called Pytest who can execute automatized unit tests and who allowed us to check the correct execution of the code. In order to trigger the execution of the tests every time that we pushed new code and to run them periodically, we have used Jenkins, a self-contained, open source automation server, which can be used to automate all sorts of tasks such as building, testing, and deploying software.

To conclude, using these two technologies, we obtained a full automated testing for our Python 3 Simulator.

Cloudant API

This module is related to the testing of the NodeJS API who manages the Cloudant database. Due to the fact that NodeJS is the language that we have used to build the API, we have used a testing framework called Mocha and an assertion library called Chai, explained at the technology section, to create a big set of unit tests that will verify the correct execution of every function of the API. Unlike the Simulator testing part, the Cloudant API tests have to be executed manually, as we do not have used an automation server who will run the tests for us periodically or in a triggered way. Nevertheless, all the tests were executed using just one instruction.

So, using Mocha and Chai, we could code enough unit tests that can be executed every time that we need to verify the correct execution of the whole Cloudant API code.

Blockchain API

In this module we haven't coded unit tests as well as using framework to run a set of tests. The way that we have verified the correct execution of the code is using the Google Chrome browser extension called Postman. This application lets you to do HTTP request to the API, and using the reads functions, we have tested the writes functions that we have coded.

We considered using a framework to build the testing part of this module, as other parts of the project, but as we did not have a lot of functions to test, they were not complicated to check and the lack of time, we concluded to use Postman and test the functions by our own.

Front End

The front end part, like the Blockchain API module, we have not used a framework to run a set of tests. The way that we have verified the correct execution of the code is executing the Client locally and test everything from there.

We considered to verify the correct execution of the code in this manner due to the fact that the front end was simple. The majority of the functionalities that it have involves listing objects, so we considered that to build a set of unit tests would be inefficient.

10. PLANNING

10.1 Project duration

The project has taken approximately 7 months, from the end of January of 2017 until the beginning of September of 2017, the day that we have presented it to the client.

10.2 Resources

To realise this project, we have needed 3 different resources types, software, humans and materials.

Usually, the software resources are considered as material resources, but because of the nature of the project, I have thought that it is important to separate them.

10.2.1 Human resources

Two developers who have worked 35 hours per week during all the project development. They have been responsible to plan, design, develop, test and do all the works that can appear.

10.2.2 Material resources

- Working area: a comfortable place where the project can be developed. This contains some indirect resources as electricity, heat, etc.
- Laptop: to develop the project and the report without any problems.
- Local Server: in order to develop the project in a development environment.
- Sogeti High tech Servers: we have used them so as to manage all the project, from the repository to the tasks that have to be done.

10.2.3 Software resources

- SVN: to control the different versions of the project.
- Ubuntu 16.04 and Windows 10: we have used these two operating systems in order to develop the project.

- Sublime Text 3: used to code the project.
- Jenkins: the application who has executed some unit tests.
- TeamForge: used to manage the project.
- Pages: to write the project report.
- Postman: to test the API.
- VirtualBox: to create an Ubuntu virtual machine on Windows.
- Skype for Business: to make video calls.
- Microsoft Outlook and Gmail: company and personal email service, respectively.
- IBM Bluemix: to deploy the APIs, Blockchain and Cloudant database on the internet.
- Docker: to deploy the Blockchain locally.

10.3 Iterations

Before we started to divide the tasks along the time, we had to consider that the project would be developed applying Scrum agile methodology. So, the planning is composed by a set of user histories that has to be as many independents between them as we can, so as to parallelise the software development.

As we have seen in project duration section, this project have taken from the end of January to early September. For this reason, we will see how we distribute the tasks during this period of time.

The initial planning is the first thing that we had to do. This process is where the requirements and objectives are analysed, in order to plan the tasks and estimate the costs.

So, as a result of the initial planning, we considered the main tasks. They involve the project specification, the Scrum initialisation, eight different sprints and the final stage.

The project specification and the Scrum initialization tasks have taken around one month. At the project specification, we have defined the system architecture, the class diagram, we have written an API documentation and we have introduced ourselves into the Blockchain. Then, during the Scrum initialization task, we have analyzed the requirements, in order to write the user histories, where we will make a first Backlog version, sorting the different user stories by the order of importance and where we will setup the development environment.

In order to have a possible product ended at the end of the each sprint, we have done three weeks long sprints.

Sprint 1 (13/03 - 31/03): at the first iteration, the development team has developed the Simulator.

Sprint 2 (03/04 - 21/04): at the second iteration, we have developed the IBM Cloudant API and a Client to manage the whole information that the system stores.

Sprint 3 (24/04 - 12/05): at the third iteration, we have integrated the Simulator, IBM Cloudant and the Client.

Sprint 4 (15/05 - 02/06): at the fourth iteration, we have learned about the Hyperledger Fabric. It's been a full iteration due to the complexity.

Sprint 5 (05/06 - 23/06): at the fifth iteration, we developed our first version of the Blockchain.

Sprint 6 (26/06 - 13/07) and 7 (17/07 - 04/08): we have developed a second version of the Blockchain with more functionalities.

Sprint 8 (07/08 - 25/08): at the last iteration, we have integrated the Blockchain in the system and we have developed a second Client version, in order to manage the information stored in the chain.

Finally, at the final stage we have finished the whole project and we have presented it to the client.

10.4 Action plan

As we have said, agile methodology is based on some iterations with user stories that have an estimated level of difficulty. Due to this fact, it is possible that we have a time deviation because of a bad planning of them, so we have to provide to our project an action plan to deal this problem, without affecting seriously to the initial plan.

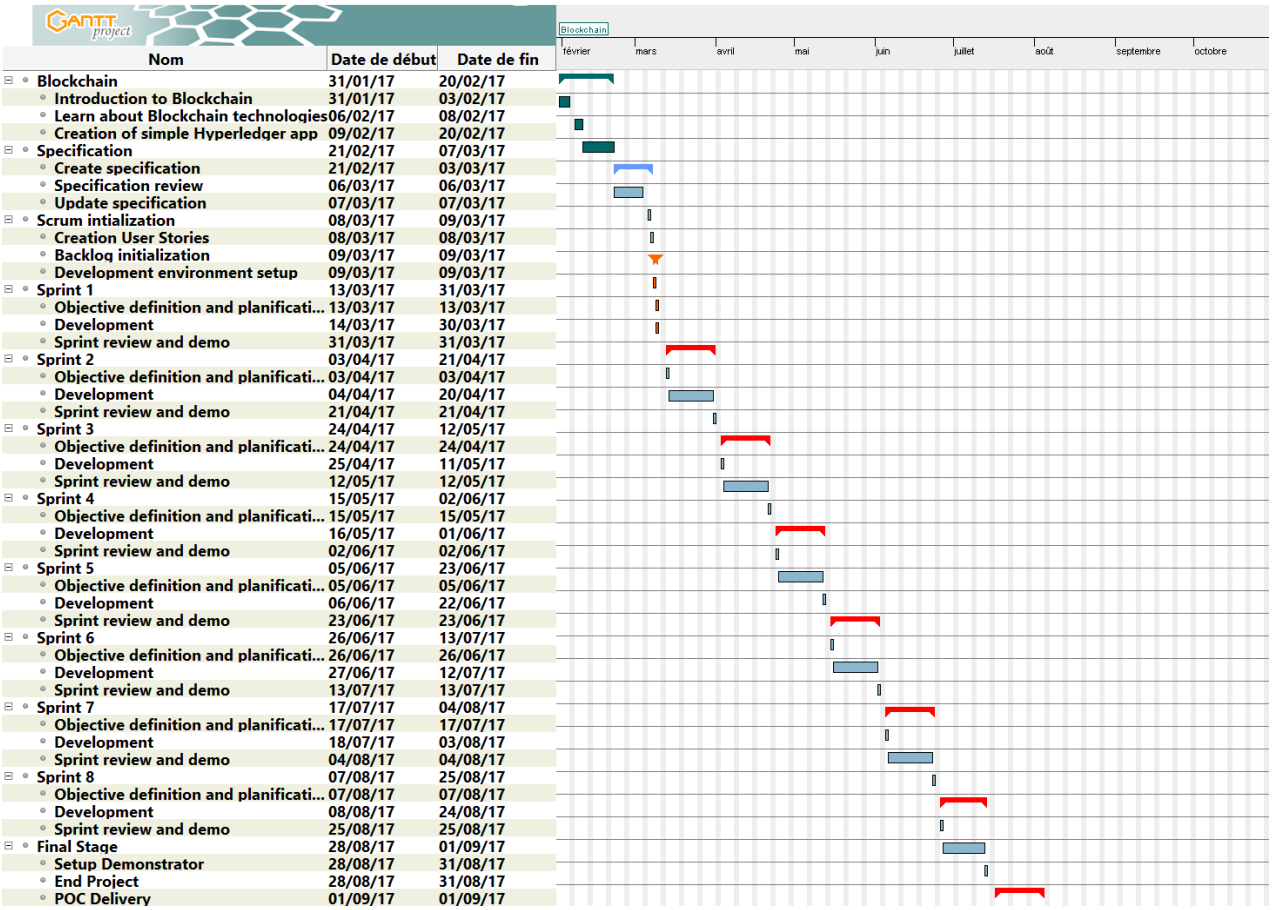
In our project, at the scenario that we have defined incorrectly the level of difficulty of the user stories, we can have two different cases.

On one hand, we have those user stories that we thought they had a bigger level of difficulty. This case has not carried any problem, as we have taken the next user story and start to work earlier.

On the other hand, we had to work more hours to finish those stories that are harder than we thought and to finish the project before the 1st of September.

To conclude, this action plan worked correctly, and following it we could finish the project before the deadline.

10.5 Gantt Diagram



Nom	Date de début	Date de fin
Blockchain	31/01/17	20/02/17
Introduction to Blockchain	31/01/17	03/02/17
Learn about Blockchain technologies	06/02/17	08/02/17
Creation of simple Hyperledger app	09/02/17	20/02/17
Specification	21/02/17	07/03/17
Create specification	21/02/17	03/03/17
Specification review	06/03/17	06/03/17
Update specification	07/03/17	07/03/17
Scrum intialization	08/03/17	09/03/17
Creation User Stories	08/03/17	08/03/17
Backlog initialization	09/03/17	09/03/17
Development environment setup	09/03/17	09/03/17
Sprint 1	13/03/17	31/03/17
Objective definition and planificati...	13/03/17	13/03/17
Development	14/03/17	30/03/17
Sprint review and demo	31/03/17	31/03/17
Sprint 2	03/04/17	21/04/17
Objective definition and planificati...	03/04/17	03/04/17
Development	04/04/17	20/04/17
Sprint review and demo	21/04/17	21/04/17
Sprint 3	24/04/17	12/05/17
Objective definition and planificati...	24/04/17	24/04/17
Development	25/04/17	11/05/17
Sprint review and demo	12/05/17	12/05/17
Sprint 4	15/05/17	02/06/17
Objective definition and planificati...	15/05/17	15/05/17
Development	16/05/17	01/06/17
Sprint review and demo	02/06/17	02/06/17
Sprint 5	05/06/17	23/06/17
Objective definition and planificati...	05/06/17	05/06/17
Development	06/06/17	22/06/17
Sprint review and demo	23/06/17	23/06/17
Sprint 6	26/06/17	13/07/17
Objective definition and planificati...	26/06/17	26/06/17
Development	27/06/17	12/07/17
Sprint review and demo	13/07/17	13/07/17
Sprint 7	17/07/17	04/08/17
Objective definition and planificati...	17/07/17	17/07/17
Development	18/07/17	03/08/17
Sprint review and demo	04/08/17	04/08/17
Sprint 8	07/08/17	25/08/17
Objective definition and planificati...	07/08/17	07/08/17
Development	08/08/17	24/08/17
Sprint review and demo	25/08/17	25/08/17
Final Stage	28/08/17	01/09/17
Setup Demonstrator	28/08/17	31/08/17
End Project	28/08/17	31/08/17
POC Delivery	01/09/17	01/09/17

Figure 12 : project tasks Gantt diagram

10.6 Found difficulties

As we have said at the Scope, the difficulties found during the project development have been mainly caused by the lack of knowledge. We have worked in a completely new environment with some technology that we have never worked with before.

In order to introduce the different obstacles, I will group the problems sorting them by stages.

State of art

Found difficulties

The main difficulties that we found was the research for similar projects. As the Blockchain technology is really young, it was difficult to find interesting projects to analyze the market.

Obtained result

After a big research, we succeeded to find some projects that have reaffirmed the utilization of the Blockchain technology in our project.

Simulator

Found difficulties

We had two different types of difficulties, in one hand we had the theory problems where we had to understand how the French rail network are built and know the common trains' consumption, in order to have accurate energy data simulated (D1).

On the other hand, we had two technical problems. Firstly, we had to deal with the software architecture problem (D2), the software has to be flexible and easily adaptable to the changes that will come in the future. Secondly, we have to provide a robust solution that ensures the correct execution of the code with every entrance (D3).

Solution

- D1: considering that all the theoretical information about the French rail network and the SNCF trains that we have compiled is correct, we have created some random entrance configuration for our simulator.

- D2: bearing in mind the fact that to build an UML and that the sequence diagrams makes you think having a bigger global perspective, we have decided to apply it to solve the difficulty D2.

- D3: considering the hypothesis that the unit tests are sufficient to verify the correctness of a function. We have solved this problem using the Pytest framework, who let us to create tests for Python code. So, we created a large number of unit tests to check the robustness of every function of the simulator.

Obtained result

The obtained results are good. We have modified some modules after having the entire simulator done, and it hasn't been difficult, as it was well-designed. Referring to the D3, we haven't had any problems with the simulator, so we can say it's really robust.

Cloudant database and API

Found difficulties

In this part of the project, we had two different types of difficulties.

On one hand, we had to learn how IBM Cloudant technology works, as no one had worked with it before (D1).

On the other hand, we had two technical problems. Firstly, we had to deal with the design of the database (D2), it has to be efficient. Secondly, we have to provide a robust API that ensures the integrity and accessibility of the data (D3).

Solution

- D1: in order to solve this lack of theoretical knowledge we have considered the hypothesis that all the sources that we have used to learn about IBM Cloudant are correct. So, we have used the official documentation of Cloudant as well as other sources where we could find useful information about this technology.

- D2: bearing in mind the that designing a database schema gives you a bigger global perspective, we have realized one in order to solve the second difficulty.

- D3: considering the hypothesis that a set of unit tests are sufficient to verify the correctness of a function, we solved this problem using two JavaScript libraries called Mocha and Chai who let us to create unit tests. So, we coded a large number of unit tests to check the correctness of the API code.

Obtained result

The obtained results are good. The unit tests helped us a lot, using them we could find lots of errors and it's been useful when we added more features to the database, including modifications to the database design.

Blockchain database and API

Found difficulties

Like other sections, we had three different types of difficulties.

On one hand, we had to learn everything related to the Blockchain and Hyperledger Fabric technology. We had to learn how the permissions, security, smart contract works, how the different elements "talks" between them, how to access to the chain as well as the technologies that we had to use to deploy it locally and generate everything that it uses (D1).

On the other hand, we had two technical problems. The first one was to add more organizations to the Blockchain, per default, you just have two organizations with two and one department, respectively (D2). The second difficulty that we had was to understand how to generate the authority certificates, used to get the permissions to communicate with the Blockchain (D3). Finally, we had to learn the Go language, so as to create the Chaincode (D4).

Solution

- D1: in order to solve this lack of theoretical knowledge, we have considered that all the sources that we have consulted to learn about the Hyperledger technology are correct. At this case, we

have used the official documentation of Hyperledger Fabric, as well as the official Hyperledger developers chat and other sources where we could find useful information about this technology.

- D2, D3 and D4: bearing in mind the hypothesis that all the sources that we have consulted are right and the correctness of every tutorial that we have followed. We have solved this difficulty reading the documentation, doing tutorials, looking and understanding some samples and finally, asking to the official developers Hyperledger chat, where we got a useful hand.

Obtained result

The obtained results are good. It has been hard to develop this part, but finally everything worked. Nevertheless, we are limited by the technology. Even the last Hyperledger Fabric release works pretty good, Hyperledger Fabric v1.0 is still too young and there are some features that are still missing, for example, we can't add more organizations to a peer who has been already initialized.

Client

Found difficulties

At this part of the project, we just had problems with the Angular 2 framework, due to the lack of knowledge (D1). That was given by the fact that it was the first time that we worked using this technology.

Solution

- D1: in order to solve this lack of theoretical knowledge we have considered that all the sources that we have consulted to learn about the Angular 2 technology were correct. At this case, we have used the official documentation of Angular 2, as well as other sources where we could find useful information about this technology and libraries.

Obtained result

The obtained results are good. Due to the fact that it was not the first client that we have developed, we did not have lots of problems.

Integration

This section is related to the integration of all the different modules of the project to one, we had to connect the Simulator, the two databases and the client.

Found difficulties

One of the most important Blockchain problems, is the time that this technology takes to add a new block to the chain. In our case, it has affected at the time that we have sent the energy data from the train, simulated by our simulator, to the database.

On one hand, we have the IBM Cloudant from IBM Bluemix which does not take a long time to save a document. Nevertheless, on the other hand we have the Blockchain. We will use the Hyperledger Fabric Blockchain implementation and, even it is not the slowest, it takes more time than IBM Cloudant to add the energy data and send the response to the client. The problem that it brings us is that it'll be inefficient to wait for the answer from the Blockchain API, as the CloudantDB will have added the piece of information before (D1).

Then, the second technical issue that we had is to control the saving data errors, we have that the Simulator sends the data directly to our API, who is going to save the data to the Blockchain and to the CloudantDB. This is not efficient, firstly for the D1, secondly, we don't have an independence of the two APIs and, finally, we make to the train store the energy data until the API returns that everything is correctly saved (D2).

Solution

- D1: Considering the hypothesis that would be better to have two different queues for each technology, we have decided to use an open standard application layer protocol for message-oriented middleware called Advanced Message Queuing Protocol (AMQP). The

main features of this technology are message orientation, queuing, routing, reliability and security.

In this case, we will use the RabbitMQ implementation, as it extends the IBM Bluemix services and it will allow us to have two different queues, one for the Blockchain messages, and another one for the CloudantDB.

So, using this architecture, we won't have the technical problem that we have said in the previous section, as it will help us to achieve the sending data independence and the saving time speed of the Blockchain won't affect to the CloudantDB.

- D2: this difficulty has been solved using a functionality that RabbitMQ technology has, which allows us to put again the element in the queue. So, bearing in mind the hypothesis that Cloudant and Blockchain API works correctly, we can solve the problem adding the element again to the queue when the status code from the APIs says that something has gone wrong.

  Going deeply, RabbitMQ allows us to use an ACK option, who says if the message has been correctly treated, or not. So, if there's any problem, we just throw the noACK option and the RabbitMQ will put the element to the queue once again, otherwise, we throw the ACK option, and the element will be deleted. In order to know if there was an error, we just need to use the returned API status.

Obtained result

The obtained results are good. RabbitMQ technology, and its own functionalities, has fitted perfectly to our difficulties. We can conclude that It would have been really difficult to solve these problems without RabbitMQ.

11. COSTS IDENTIFICATION

As in every project, it is important to do a project estimated costs analysis, in order to calculate the initial budget and to plan the different expenses.

11.1 Estimation of costs

In order to estimate the costs, I have divided them into two big groups, the human and material resources costs.

11.1.1 Human resources costs

As it is explained at the temporal planning, the project has been realized by two developers. However, we have done some hours out of Sogeti High Tech bureau, in order to finish the final report.

As a consequence of this division, the human costs are:

	Estimated hours	Price per hour (€)	Number	Total (€)
Developers bureau hours	840	5.71	2	9592.8
Developers hours out from bureau	100	5.71	1	571
Project director	120	21	1	2520
Human resources	15	14	1	210
Social Security	2%	percentage	2	191.86
Total estimated	1780		7	13085.66

Figure 13 : Human resources costs table

11.1.2 Material resources costs

The material resources costs are these expenses that are directly related to the infrastructure and the software.

	Units	Price (€)	Useful life	Total estimated amortisation (€)
Dell Latitude E5470	2	839	4 years	207.5
Sublime Text	2	53.45	4 years	13.36
Oracle VirtualBox	2	50	4 years	12.5
Microsoft Office	2*6	9.99	1 month	119.88
Microsoft Windows 10 Professionnel	2	50.82	4 years	12.71
Ubuntu 16.04	2	0	Unlimited	0
Jenkins	1	0	3 months (Trial version)	0
Pages	1	0	Unlimited	0
Skype for Business	2*6	150	6 months	150
Gmail	1	0	Unlimited	0
IBM Bluemix	2*4	200	4 months	200
Docker	2*4	0	Unlimited	0
Internet, bureau, electricity, etc	6	700	1 month	4200
Public transport	2*6	15	1 month	180
Total estimated		8073.45		5095.95

Figure 14: material resources costs table

To conclude, we can see that both estimated costs bring us the total of 18181,61€.

11.2 Control management

The calculated costs do not allow any deviation or error, therefore, is calculated that everything goes as we have planned. If we had had this scenario, we had not stopped working, we would have tried to improve the software and add those functionalities that we had dismissed before.

Nevertheless, we have to estimate the costs of the deviations that we can have while we develop the project and add it to the estimated budget. So, as we said, the main problem that we could have had during the project development is the time deviation, however, we could have had other deviations that are much less probable.

11.2.1 Time deviation

If we take a look at the action plan section, we could end the last iteration without achieving the defined objectives, so what we planned to do solve this deviation is to consider extra hours. In this case, we should do some of this extra work at the office, so the budget could increase around 10% of the hours strictly dedicated to the project. So, the total costs could increase (840 hours * 0.1 * 2 developers * 5.71€/hour) + (840 hours * 0.1 * 1 developer * 0€/hour) = 959.28€.

11.2.2 Other deviations

Apart from the time deviation, we could have had some others deviations that could be important to note. The most evident problem would be to have a breakdown at the computers. As we should buy a new computer, we should add 207.5€ to the budget.

Last but not least, to ensure other expenses that cannot be prevented, we should add a percentage of contingencies of 5% of the total cost to ensure a reliable budget that we know we're not going to overcome.

11.3 Total estimated budget

As a result of the deviations and the project cost, the estimated project budget will be 20315,8095 €, as we can see in the Figure 15.

Resources	Cost (€)
Human resources	13085.66
Material resources	5095.95
Time deviation	959.28
Other deviations	207.5
Contingencies	5% of the total
Project budget	**20315.8095**

Figure 15: Total costs table

12. SUSTAINABILITY

12.1 Economic dimension

In order to consider the economic viability, we have to evaluate the different costs and the materials and human resources.

So, to realise this project, we have tried to estimate the costs as well as we could. Then, if the client that we will present the project wants to integrate it into his system, the enterprise Sogeti High Tech will earn lots of benefits. Apart from the initial product cost, the client will have to pay the maintenance of the software, and, due to the fact that this application will supply the whole French rail network and will involve some enterprises, the benefits are going to be high. Finally, if this project works correctly, the enterprise can present this to other clients in order to provide them solutions using the technologies that we have used.

In conclusion, at this dimension, the project deserves 18 points. Due to the fact that the project is considered viable and the risks are well defined. As we can see, this is not the maximum note, due to the fact that the project could be done in less time. That could be possible if they had hired two developers with more experience or more developers. However, both scenarios would have increased the budget.

12.2 Social dimension

This project will have a French social impact, If the client wants to integrate it into his system. This consequence, is because the software will cover a future requirement that the whole French rail networks will have. As it is explained in the introduction, this new application will allow them to share and track the energy consumption of the different train companies, in an easy, transparent and secure way, without having to make a big initial effort.

Apart from that, as this software may just promote to France and the enterprises that will use the French rail networks, we could say that it won't deeply improve the society, at least, in a short time. This might happen in the case that the application scale to many other countries.

To conclude, at the social dimension, the project deserves 15 points, as this project will cover the future requirement, but there will be more useful functionalities to add and it is just focused to France.

12.3 Environmental dimension

Before we started the project, Sogeti High Tech had already the material resources that we will use to develop it, like the server or the laptop, so it won't be necessary to buy new ones. Apart from that, the electricity consumption of them is not significative, as the server will be shared.

In addition, we will take profit of other projects that have been done before, so we will save resources, time and money. About the product, as it is just software, it won't have any costs of fabrication and it won't have any environmental impact. Then, we do not have to worry about the recycle, we will just delete the project from the hard drive.

So as to add more information, we will try to make a good electrical and heat management, at the bureau, in order to use the minimum resources that we can. Some examples could be to turn off the computer every time that we will not use it more than 1 hour, not to set the heat too strong, more than 23 degrees, etc. Apart from that, we will always use the technology that we have in order to save resources, for example, we won't print documents that we can read using our computers.

Because of these aspects, the project deserves 15 points at the environmental dimension.

12.4 Sustainability table

To conclude, we can see in the Figure 16 the results of the sustainability study that I have explained.

Sustainability	Economic	Social	Environmental
Plan	Economic viability	Better life quality	Resources analysis
Points	18	15	15

Figure 16: Sustainability table

13. CONCLUSIONS

13.1 Achieved goal

At the beginning of this paper, there was one specific goal that this project was focused on. Develop an application that allows to the clients, who use the French rail network, manage their trains energy waste, in a safe, transparent and accurate way.

So, after developing the application, we can say that this goal has been correctly accomplished. Using our application, the different companies can get the amount of energy consumed by their trains. This data is the one who has to be safe and transparent, so, thanks to the Blockchain technology, and his physic architecture, we can ensure that the data can not be modified, and consequently, safe. Then, as everyone, that have the correct permissions, can read what is written in the Blockchain, we can ensure the transparency that the companies want.

13.2 Future extensions

This project is just the beginning of the proof of concept. This project will continue to be developed for a long time. So, in order to add more value to it, we have thought some interesting functionalities that would be useful to the clients.

The first functionality is to implement the auditor role, who have the permissions to access to every piece of information that the two databases contains.

Another interesting option that will be added, is the consumed energy payment, we will allow for the companies to pay directly through our application. This functionality will be directly related to the ERESS organisation, the company who will invoice the consumed energy.

Finally, the team will have find out how to deal with the time that the Blockchain needs to add a new block. There is a new type of transactions called Transaction Family who is faster, so, in the future, the developers will put under analysis this new transaction system in order to see if they can implement it in our application.

13.3 Personal evaluation

Thanks to Sogeti High Tech and the opportunity that they gave to me, I have learned in a way that I did not expect. There are a big set of other things that I have learned, apart from the ones related to the project.

Firstly, the internship gave me the opportunity to work in a completely different environment. I have worked in a French team, where I have felt integrated, in spite of that I had difficulties to speak a good French, and where I have learnt how to deal with people who do not come from the same place as you. Even though France is not that far from Spain, they are really different, they life their lives in a completely different way.

Secondly, I have learned about the company heritage and management. As this has been the first time that I have worked in a such a big company as Sogeti High Tech, we do not have to forget that is a filial from Capgemini.

Thirdly, as I have said previously, I have learned about the development process of Python applications, Angular 2 clients, Hyperledger Fabric Blockchains and IBM Cloudant databases. Where showed me the importance that the Blockchain technology will have in this world in a near future.

Finally, to sum up everything, I can conclude that this 6 months have been really interesting and profitable and an experience that I would recommend to everybody.

14. BIBLIOGRAPHY

[1] sogeti-hightech.fr, (n,d). Sogeti High Tech - High Tech & Global Product & Engineering Services. [online] Available at: https://www.sogeti-hightech.fr [Accessed 28 Feb. 2017]

[2] capgemini.com, (n,d). - Capgemini - Capgemini Consulting Worldwide. [online] Available at: https://www.capgemini.com [Accessed 28 Feb. 2017]

[3] sncf.com, (n,d). - SNCF - Trains, Services, Entreprises, Emploi. [online] Available at: http://www.sncf.com [Accessed 28 Feb. 2017]

[4] alstom.com, (n,d). - Alstom. [online] Available at: http://www.alstom.com [Accessed 7 Oct. 2017]

[5] renfe.com, (n,d). - Renfe. [online] Available at: http://www.renfe.com [Accessed 7 Oct. 2017]

[6] arxiv.org, (n, d). Intelligent Vehicle-Trust Point: Reward based Intelligent Vehicle Communication using Blockchain. [Online] Available at: https://arxiv.org/ftp/arxiv/papers/1707/1707.07442.pdf [Accessed 13 Aug. 2017]

[7] pdf-archive.com, (n, d). Blockchain enabled Trust & Transparency in supply chains. [Online] Available at: https://www.pdf-archive.com/2017/02/01/project-thesis-anders-j-rgen/project-thesis-anders-j-rgen.pdf [Accessed 13 Aug. 2017]

[8] pwc.com, (n, d). Blockchain – an opportunity for energy producers and consumers? . [Online] Available at: https://www.pwc.com/gx/en/industries/assets/pwc-blockchain-opportunity-for-energy-producers-and-consumers.pdf [Accessed 13 Aug. 2017]

[9] agilemethodology.org, (n,d). The Agile Movement. [Online] Available at: http://agilemethodology.org [Accessed 28 Feb. 2017]

[10] Agilemanifesto.org, (2016). Manifesto for Agile Software Development. [online] Available at: http://www.agilemanifesto.org/ [Accessed 28 Feb. 2017].

[11] Kanbantool.com, (2016). Kanban Methodology | Kanban Tool. [online] Available at: http://kanbantool.com/kanban-methodology [Accessed 28 Feb. 2017].

[12] scrum.org, (2008). What is scrum?. [online] Available at: https://www.scrum.org/resources/what-is-scrum [Accessed 28 Feb. 2017].

[13] subversion.apache.org, (2000). Apache Subversion. [online] Available at: https://subversion.apache.org/ [Accessed 23 Mar. 2017].

[14] chat.hyperledger.org, (2008). Hyperledger Chat. [online] Available at: https://chat.hyperledger.org/home [Accessed 21 Aug. 2017].

[15] eress.eu, (n,d). Eress. [Online] Available at: http://eress.eu [Accessed 29 Aug. 2017]

[16] cisco.com, (n,d). Cisco - Internet of Things (IoT). [Online] Available at: http://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html [Accessed 28 Feb. 2017]

[17] python.org, (n, d). Python 3.6. [Online] Available at: https://www.python.org [Accessed 30 Aug. 2017]

[18] docs.pytest.org, (n, d). Pytest: helps you write better programs. [online] Available at: https://docs.pytest.org/en/latest/ [Accessed 21 Aug. 2017].

[19] jenkins.io, (n, d). Jenkins. [online] Available at: https://jenkins.io [Accessed 21 Aug. 2017].

[20] rabbitmq.com, (n, d). Pytest: helps you write better programs. [online] Available at: https://www.rabbitmq.com [Accessed 2 Sep. 2017].

[21] mongodb.com, (n, d). What is MongoDB?. [Online] Available at: https://www.mongodb.com/what-is-mongodb [Accessed 15 Aug. 2017]

[22] ibm.com, (n, d). IBM Cloudant. [Online] Available at: https://www.ibm.com/analytics/us/en/technology/cloud-data-services/cloudant [Accessed 15 Aug. 2017]

[23] couchdb.apache.org, (n, d). Introduction. [Online] Available at: http://docs.couchdb.org/en/2.1.0/intro/index.html [Accessed 15 Aug. 2017]

[24] bigcouch.cloudant.com, (n, d). BigCouch. [Online] Available at: http://bigcouch.cloudant.com [Accessed 15 Aug. 2017]

[25] ibm.com, (n, d). IBM Bluemix. [online] Available at: https://www.ibm.com/us-en/marketplace/cloud-platform [Accessed 2 Sep. 2017].

[26] nodejs.org, (n, d). NodeJS. [online] Available at: https://nodejs.org/en/ [Accessed 2 Sep. 2017].

[27] ibm.com, (n, d). Into the wild BLUE yonder!. [online] Available at: https://www.ibm.com/developerworks/community/blogs/gcuomo/entry/javascript_everywhere_and_the_three_amigos?lang=en [Accessed 2 Sep. 2017].

[28] mochajs.org, (n, d). Mocha. Simple, flexible, fun. [online] Available at: https://mochajs.org [Accessed 2 Sep. 2017].

[29] chaijs.com, (n, d). Chai Assertion Library. [online] Available at: http://chaijs.com [Accessed 2 Sep. 2017].

[30] bitcoin.org, (n, d). Bitcoin. [Online] Available at: https://bitcoin.org/en/ [Accessed 15 Aug. 2017]

[31] corda.net, (n, d). Welcome to Corda !. [Online] Available at: https://docs.corda.net [Accessed 15 Aug. 2017]

[32] hyperledger.org, (n, d). About Hyperledger. [Online] Available at: https://www.hyperledger.org/about [Accessed 15 Aug. 2017]

[33] ethereum.org, (n, d). What is Ether?. [Online] Available at: https://www.ethereum.org/ether [Accessed 15 Aug. 2017]

[34] javascript.com, (n, d). JavaScript. [online] Available at: https://www.javascript.com [Accessed 2 Sep. 2017].

[35] golang.org, (n, d). The Go Programming Language. [online] Available at: https://golang.org [Accessed 3 Sep. 2017].

[36] angular.io, (n, d). Angular. [online] Available at: https://angular.io [Accessed 3 Sep. 2017].

[37] typescriptlang.org, (n, d). TypeScript the JavaScript that scales. [online] Available at: https://www.typescriptlang.org [Accessed 3 Sep. 2017].