EPITECH

Cash Manager Project

Groupe 27

Dylan

Ayoub

Thomas

Mohamed

Table of Contents

[PRESENTATION 2](#_Toc26306297)

[1. Delivred Tools : 2](#_Toc26306298)

[FUNCTIONAL SPECIFICATIONS 3](#_Toc26306299)

[1. Project Description: 3](#_Toc26306300)

[2. Software Functional Architecture: 3](#_Toc26306301)

[3. Use Cases : 4](#_Toc26306302)

[4. Impact Mapping Diagram: 4](#_Toc26306303)

[5. Tree Diagram : 5](#_Toc26306304)

[6. Activity Diagram : 5](#_Toc26306305)

[7. Application Interfaces : 6](#_Toc26306306)

[8. Features Rules : 7](#_Toc26306307)

[TECHNICAL SPECIFICATIONS 8](#_Toc26306308)

[9. Technology Choices : 8](#_Toc26306309)

[10. Software Technical Architecture : 10](#_Toc26306310)

[11. Class Diagram : 11](#_Toc26306311)

[12. Data Diagram : 12](#_Toc26306312)

[13. API Documentation : 12](#_Toc26306313)

[Payment 12](#_Toc26306314)

[Machine 12](#_Toc26306315)

[Article 13](#_Toc26306316)

[Setting 13](#_Toc26306317)

[Cart 13](#_Toc26306318)

[14. Unit Test and Code Coverage : 13](#_Toc26306319)

[15. Devops : 15](#_Toc26306320)

|  |  |  |
| --- | --- | --- |
| Nom | Version | Date |
| Dylan B | 1.0 | 14/10/2019 |
|  |  |  |

Cash Manager EPITECH

# PRESENTATION

## Delivred Tools :

The Cash Manager project is delivered as a zip file that consists of 3 folders, a back folder, a Front (Mobile) folder and a last folder for documentation.

To start the back, just go to the Back / CashManager / folder and launch the run\_docker.sh file (under linux). If you are on windows, get inspired from the run\_docker.sh file (the commands are almost the same).

To have a view of the front open the front folder in android studio .

To lunch the aplication in your phone device activate the developer mode in your phone , connect it with your computer and build the graddle to get the application opened in your phone .

In the case you don’t have an android phone download an android emulator and build the gradle ,the application will be opened in your emulator .

When the application is run in your phone you can start to test it by logging first of all after that you can add and remove article in your cart once the choice is finished you can procced to the payment and in this screen you have the choice between paying by check or by NFC (card mode ); to pay by check scan you QR-code and for card just scan the card by your phone .  
You will get a comfirmation of the payment state in both situations denied or accept.

After that you can see that you get a general view of the whole cash manager Application .

# FUNCTIONAL SPECIFICATIONS

## Project Description:

Conception of a cash register software that allows users to:

• connect to the terminal

• display the cash register

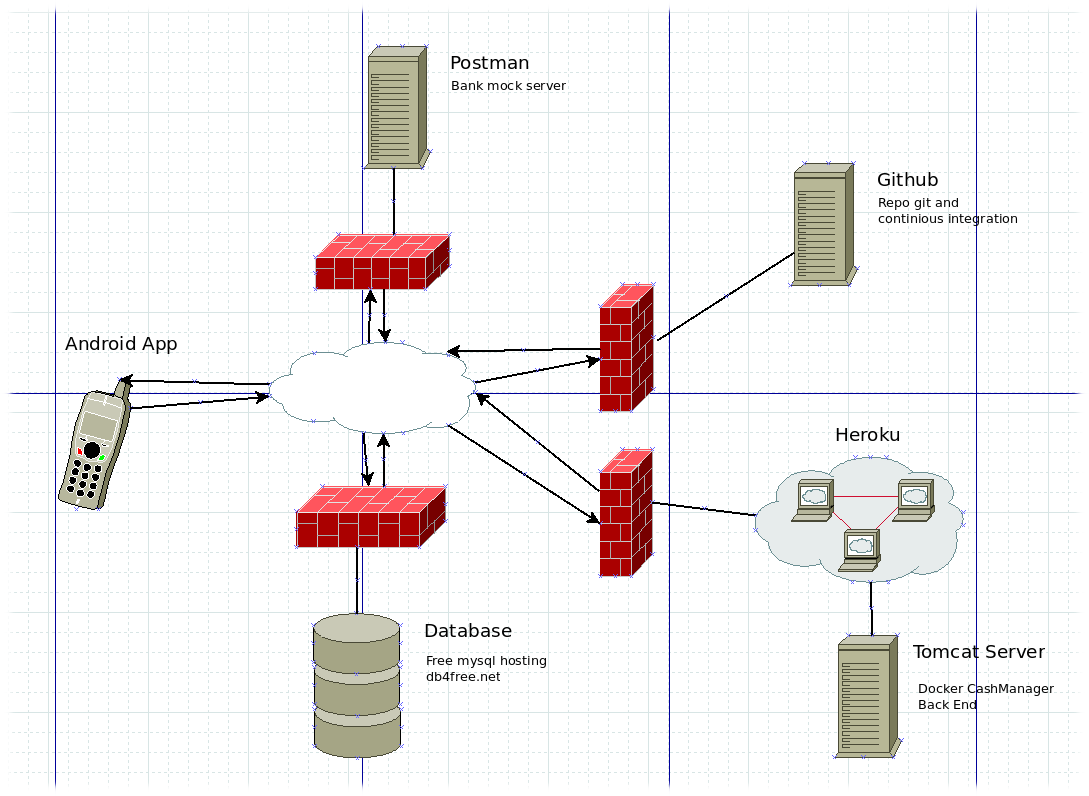
• Provide a payment method

• It allows the user to scan his check,

• it allows the user to display the total amount of his invoice,

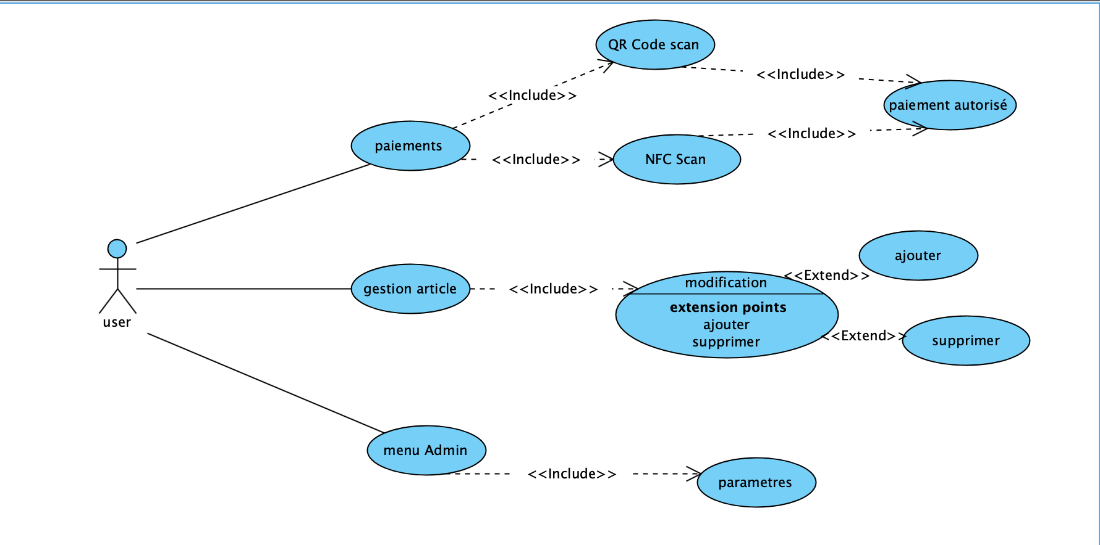
• display the control status (authorized, denied, waiting)

## Software Functional Architecture:



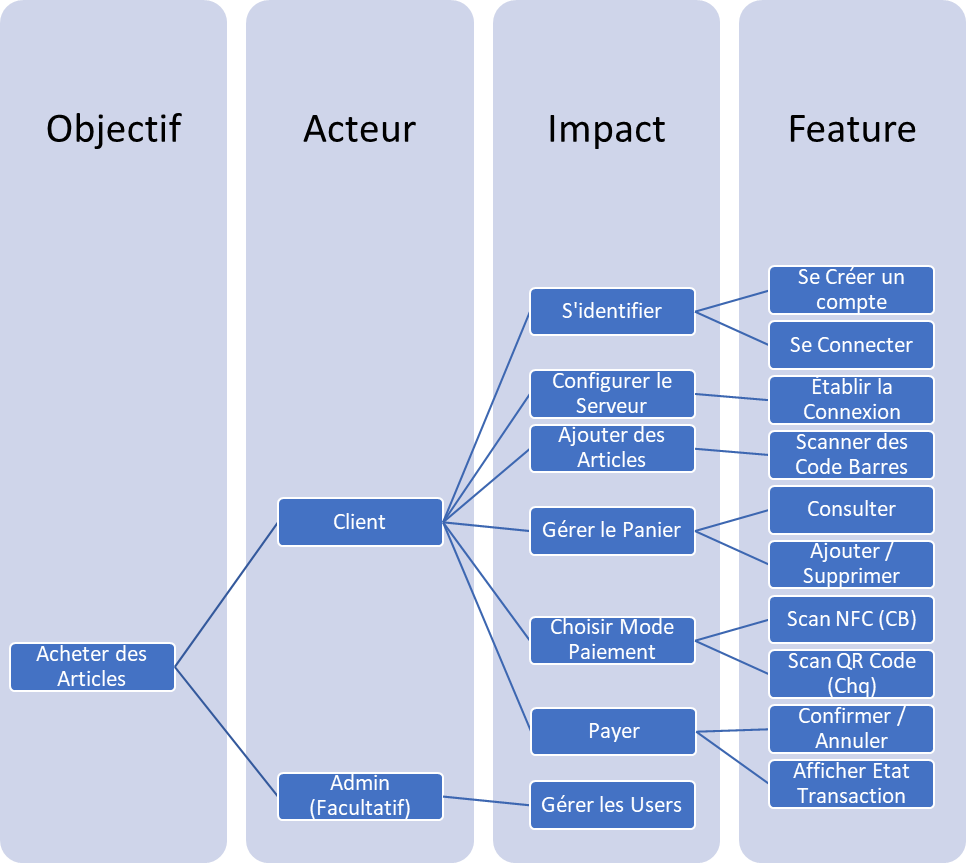
We chose to separate the server from the database because these are the only free offers we found to associate heroku with a mysql database. We used Github because it offered the features we needed (repo git and pipline for continuous integration). Postman was chosen to create a simulation of a bank server. Postman offers a free server mock deployment, that's why we chose it.

## Use Cases :



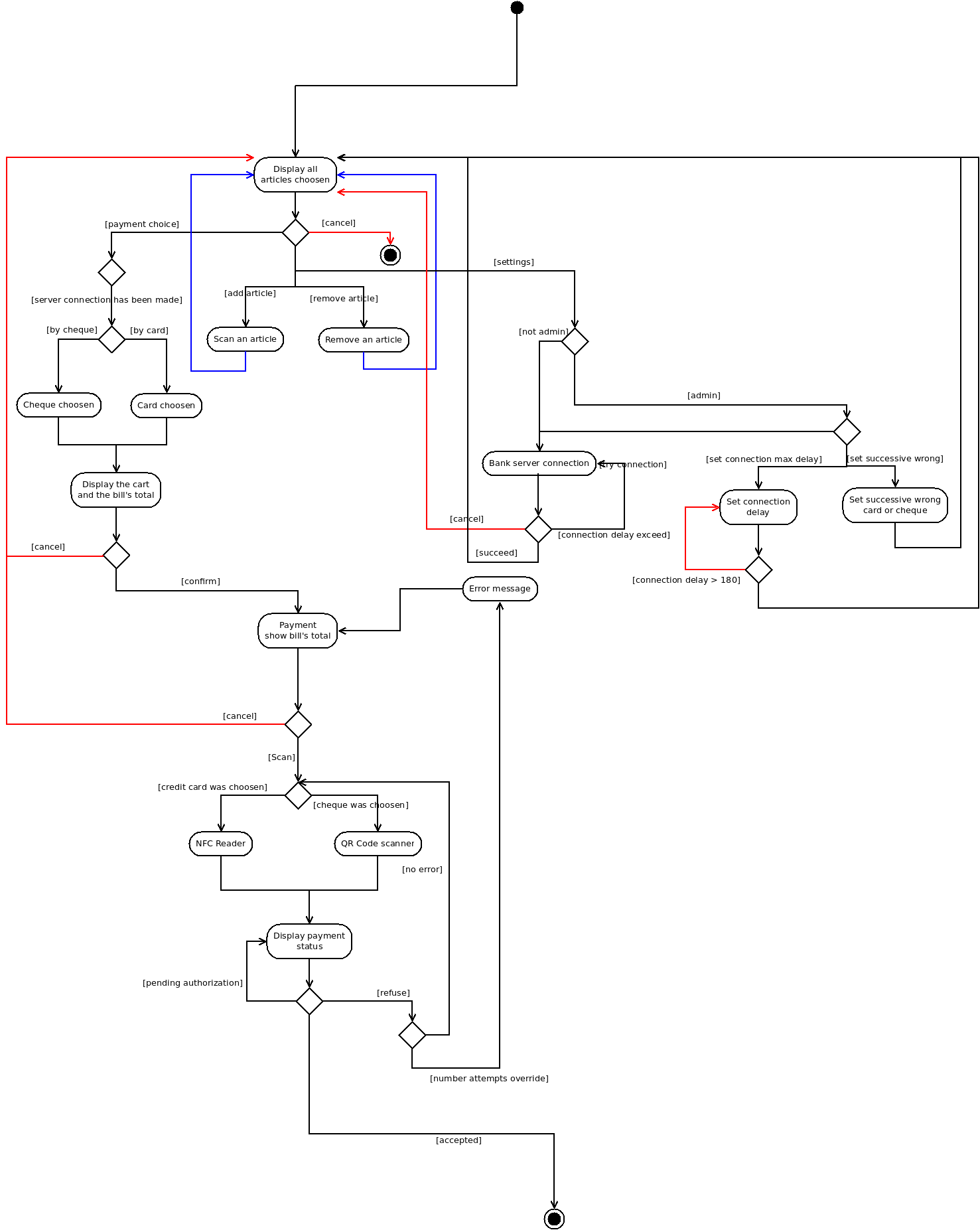
This part explain how our application is running the all part to allow the readers to understand a typical execution of the back-end of our cash manager .

## Impact Mapping Diagram:



## Tree Diagram :

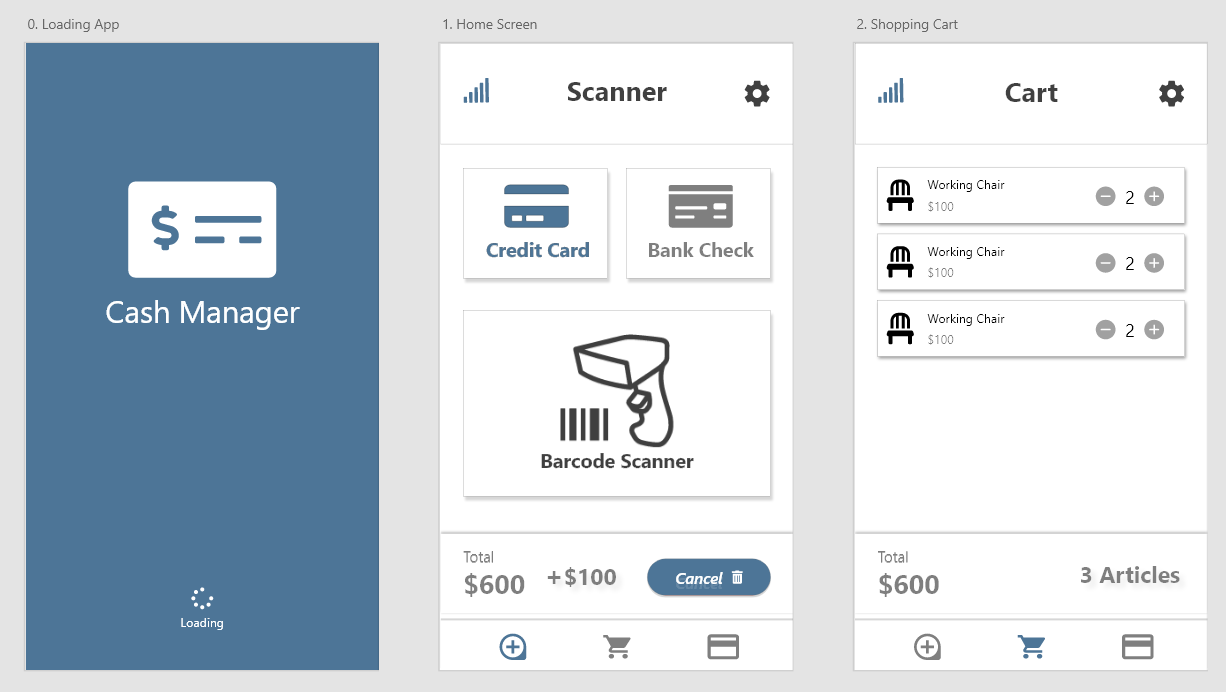
## Activity Diagram :

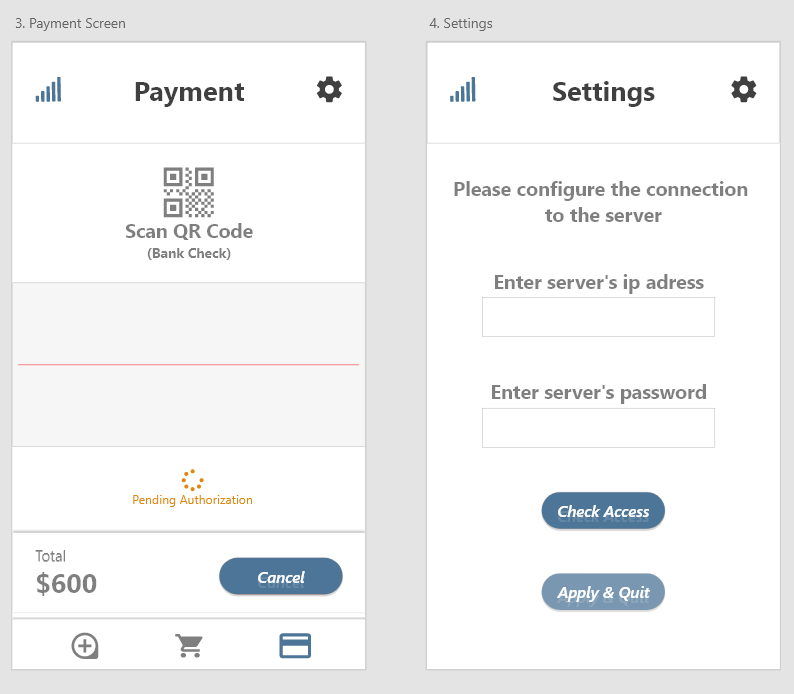


The activity diagram was generated by Uml it allows the developers to pursuit a good achievement of the coding part in a perfect environment.

## Application Interfaces :

We used AdobeXD.to create the different IHM.





## Features Rules :

|  |  |
| --- | --- |
| Features | Link Constraints |
| Create account | Linking to the database to create profiles |
| Establish connection | Find if the profile is created and connect to it if not create a new one |
| Scan the barre-code | Add items to cart as well as save them to the cart |
| Scan the NFC device | Recover credit card data to make payment |
| Scan the QR Code | Collect the give of the check to be able to cash it |
| Cancel / Confirm | Confirmation if payment is made or cancellation of payment |
| State transition screen | Check if the payment is done and the payment is not accepted. |

# TECHNICAL SPECIFICATIONS

## Technology Choices :



Java is a general-purpose, advanced, object-oriented programming language whose syntax is close to C ++. Its characteristics as well as the richness of its ecosystem and its community allowed it to be very widely used for the development of very disparate types of applications. Java is widely used for the development of business and mobile applications.



Kotlin is a language developed by JetBrains, a company that designs EDI Intellij Idea among others. However it is open-source

Its main advantages, to use the main arguments of the official site, lie in the simplification of the code for many common tasks (for example, the creation of POJO), an increased code security thanks to a stricter control than Java (especially for the nullity of the values), and an ease of interaction to and from the Java language and with the JVM - which allows in particular the direct reuse of most codes written in Java



Git is a distributed version-control system for tracking changes in source code during software development . It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed,data integrity ...﷟HYPERLINK "https://en.wikipedia.org/wiki/Data\_integrity"



The means that allowed us to work remotely on the project and to see the personal progress of each member of the project.

This tool is very fluid and useful for the work in a project where we are brought to complement each other in our work.



The dia technology s [free and open source](https://en.wikipedia.org/wiki/Free_and_open_source_software) general-purpose diagramming software, developed originally by Alexander Larsson. Dia uses a controlled [single document interface](https://en.wikipedia.org/wiki/Single_document_interface) (SDI) similar to [GIMP](https://en.wikipedia.org/wiki/GIMP) and [Inkscape](https://en.wikipedia.org/wiki/Inkscape).

It enabled us to generate the UML diagrams of our project allowing the global visualization of the project before its realization.We use it because it was a good idea to explore new technology in our project.



Adobe XD is a [vector-based](https://en.wikipedia.org/wiki/Vector_graphics_editor) [user experience design tool](https://en.wikipedia.org/wiki/User_experience_design) for [web apps](https://en.wikipedia.org/wiki/Web_app) and [mobile apps](https://en.wikipedia.org/wiki/Mobile_app), developed and published by [Adobe Inc](https://en.wikipedia.org/wiki/Adobe_Inc.). It is available for [macOS](https://en.wikipedia.org/wiki/MacOS) and [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), although there are versions for [iOS](https://en.wikipedia.org/wiki/IOS) and [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) to help preview the result of work directly on mobile devices. XD supports [website wireframing](https://en.wikipedia.org/wiki/Website_wireframe), and creating simple, immersive, interactive click-through prototypes.

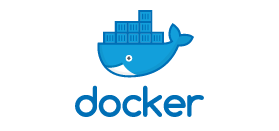
Adobe XD was used to generate the Mock-up of our mobile application because it’s useless and functional in the case of our project.



This is the technology we used for payment on our application, it allows us to retrieve information from the bank card automatically.it was one of the technology that the project said to us to work with.



Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps.we use this technolgy because it was one of the cheapest ones in the market and it’s so performent in the case of our project.



Docker allows us to embed an application in a software container that can run on any machine server, whether physical or virtual. We use the docker technology to build our constainer .We use docker because it’s one of the best technogy for the devOps problematique and it’s one of the technology we are familiarized with .

## Software Technical Architecture :

We used for the back server a DAO architecture, service, controller, entity. We were inspired by a method used to develop with the Springboot framework. Here the dao are used to communicate with the database, the controllers (named servlet in JEE) recovers the client requests, and the services are used to connect the controllers to the dao. The services also offer the possibility of verifying the validity of the data sent by the customer. In addition to this, we have added a table class that simply describes the contents of the different tables in the database (in springboot, the description of the tables is done directly in entities). We preferred to separate the entities from the tables to unload the entity file.

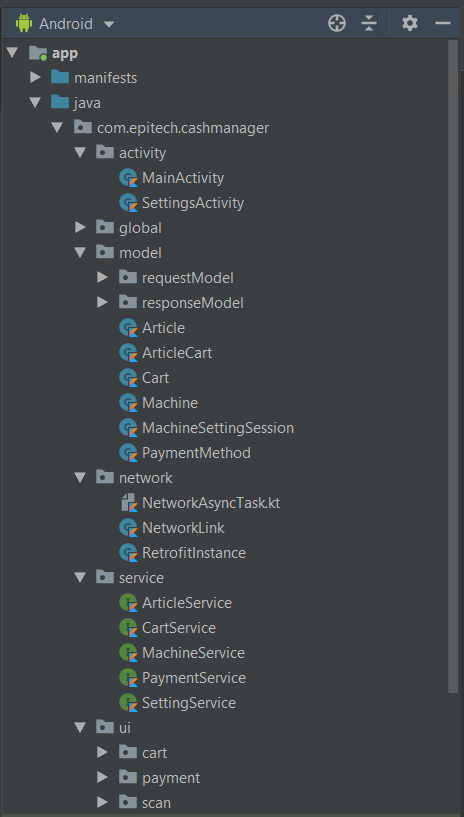
This system was chosen to facilitate table creation in the database. In some files, all that remained was to program the routes we needed.

We used several paterns designs. For the tests, as some class dao had the same tests, we set up a factory design patern. For services, as all functions go through the same function, we have factored them into an abstract factory design.

From what is the front part, the mobile in this case we were led to use the model MVC (model-view-controller) by the bias of Kotlin which is a product designed for projects in Android.

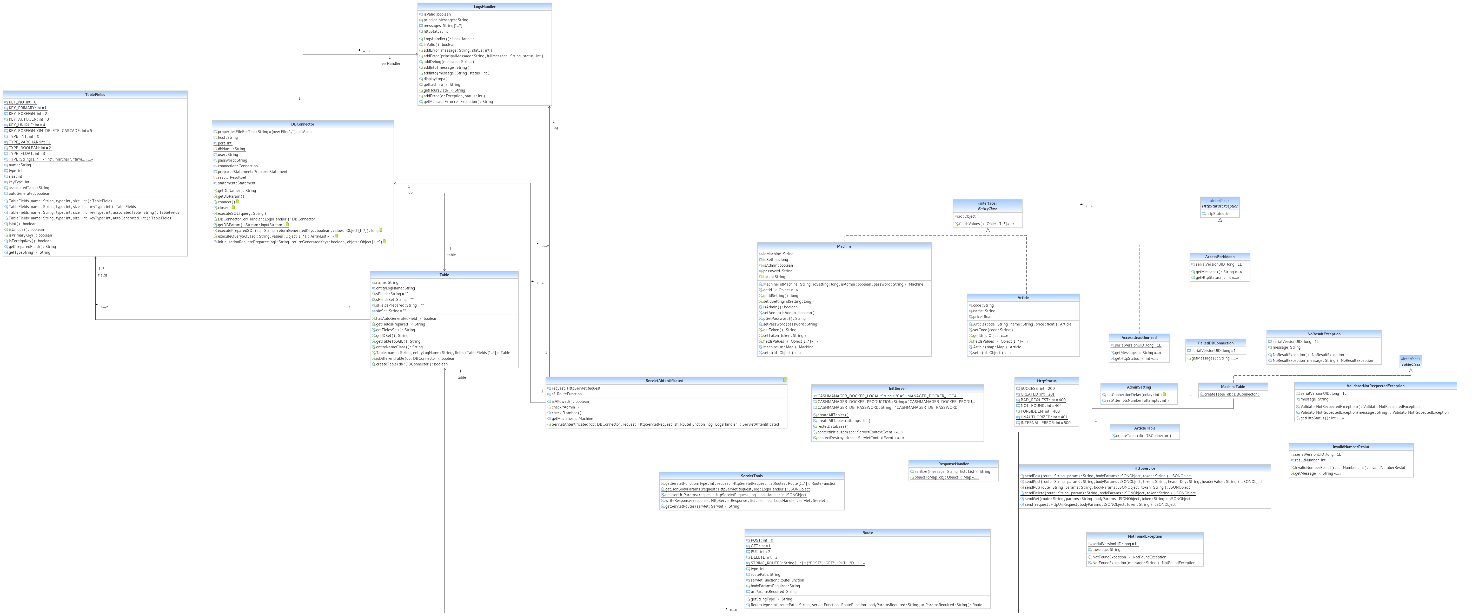
In this part, the major problems encountered are as follows: linking pages made individually to the entire project and application; Establishing the link between the front and the back to be able to call api and finally have a global view of our application and be able to put it into production for others.

*Structural View of Front Android Studio Project :*

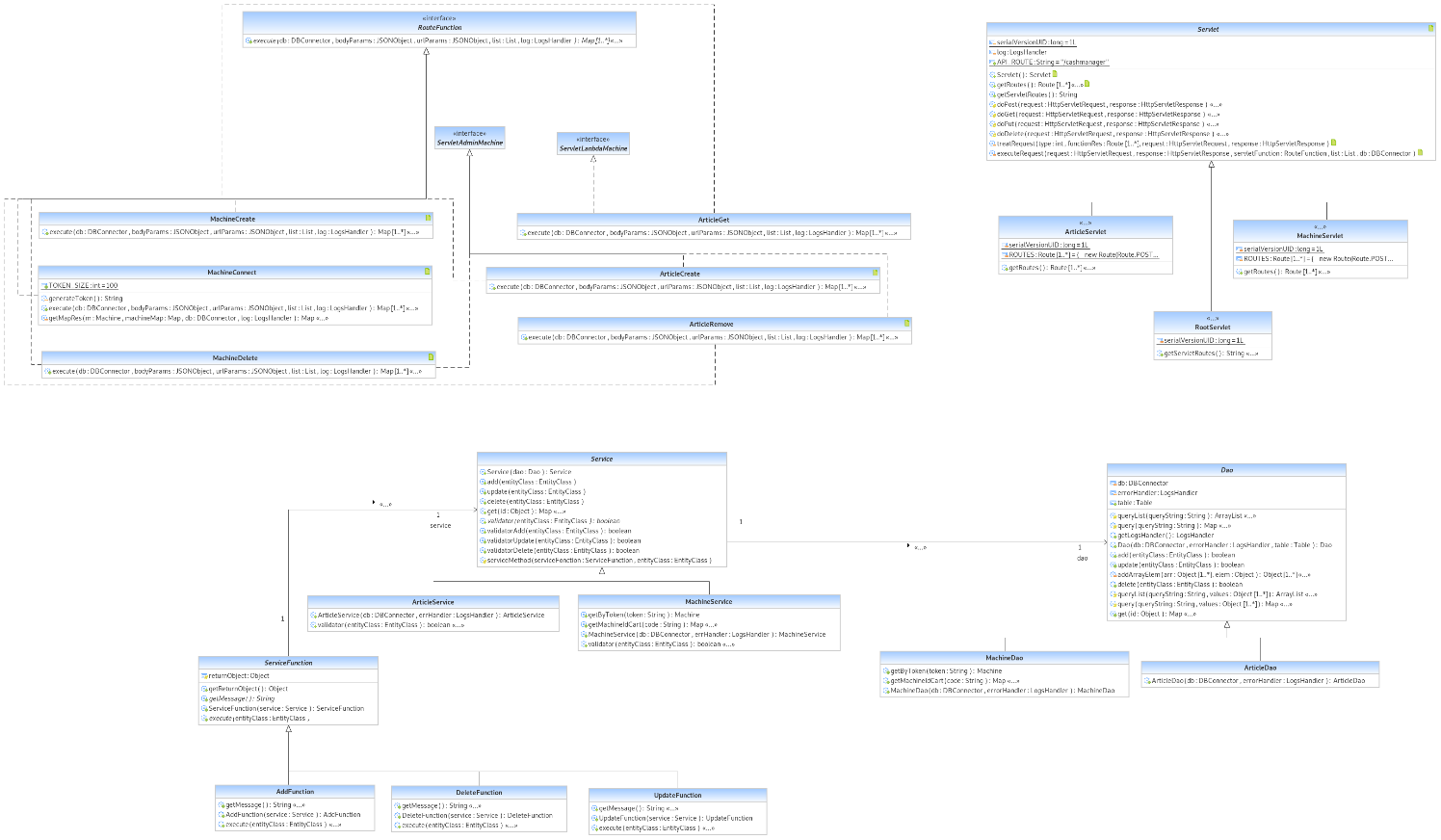


## Class Diagram :

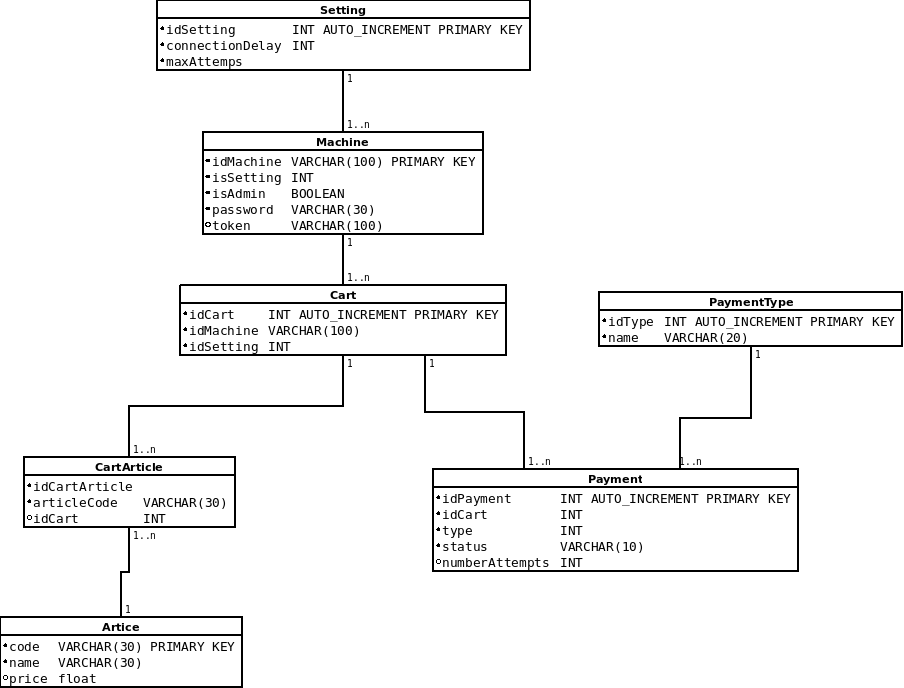
(Vous le trouverez dans le dossier Documentation/UML/diagram\_class\_CashManagerUtilsEntities.png):



(Vous le trouverez dans le dossier Documentation/UML/diagram\_class\_CashManagerDaoService.png):



## Data Diagram :



## API Documentation :

### Payment

GET /cashmanager/payment/get\_types (Machine auth token required)  
Get all payment types avaiable.

POST /cashmanager/payment/choose\_payment\_type (Machine auth token required) | Body JSON Params : int idCart, int idType  
Choose a payment type.

PUT /cashmanager/payment/pay (Machine auth token required) | Body JSON Params : int idCart | if credit card : String number, String pin | if cheque : String code  
Make the payment if the payment type is choosen.

DELETE /cashmanager/payment/cancel?idCart=? (Machine auth token required)  
Cancel the payment.

### Machine

POST /cashmanager/machine/create | Body JSON Params : String idMachine, long idSetting, boolean isAdmin, String password  
Create a machine.

POST /cashmanager/machine/connect | Body JSON Params : String idMachine, String password  
Connect the machine to the server for getting a token.

DELETE /cashmanager/machine/remove?idMachine=? (Admin machine token auth required)  
Remove a machine.

### Article

POST /cashmanager/article/create (Admin machine token auth required) | Body JSON Params: String code, String name, float price  
Create an article.

DELETE /cashmanager/article/remove?code=? (Admin machine token auth required)  
Remove an article

GET /cashmanager/article/get?code=? (Machine auth token required)  
Get an article information.

### Setting

POST /cashmanager/setting/create (Admin machine token auth required)  
Create a setting.

DELETE /cashmanager/setting/remove?code=? (Admin machine token auth required)  
Remove a setting.

### Cart

POST /cashmanager/cart/create (Machine auth token required) | Body JSON Params : String idMachine  
Create a cart.

POST /cashmanager/cart/add\_article (Machine auth token required) | Body JSON Params : long idCart, String codeArticle  
Add an article in the cart.

DELETE /cashmanager/cart/remove\_article?idCart=?,codeArticle=? (Machine auth token required)  
Remove an article from the cart.

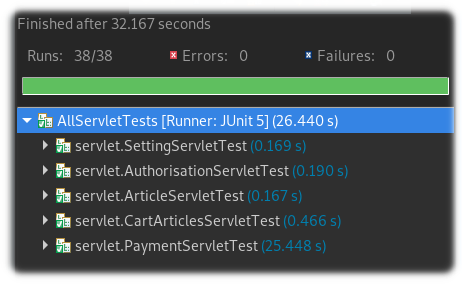
GET /cashmanager/cart/get\_articles?idCart=? (Machine auth token required)  
List all articles contained by the cart.

## Unit Test and Code Coverage :

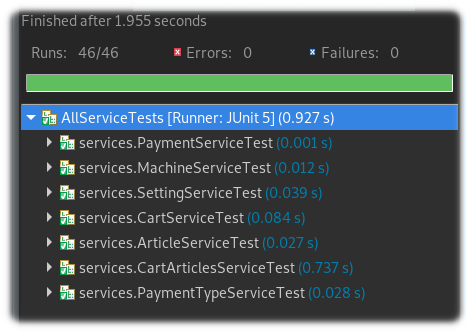
L The development of the back has been developed using unit tests. Before each deployment of each controller function or function dao / service, unit tests were created to facilitate testing of it.

A total of 84 unit tests written.

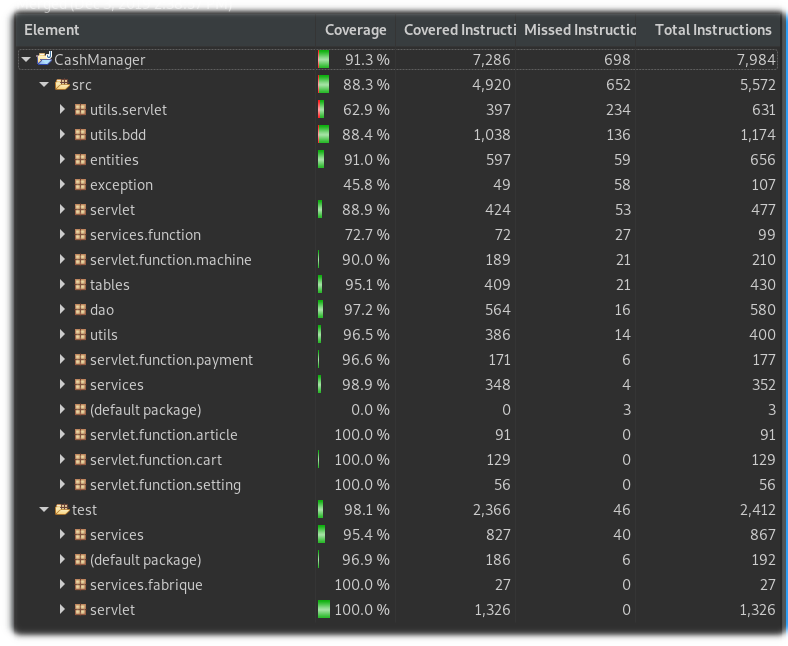
38 tests were written to test directly the controller (servlet):



46 tests were written to test directly the project tables:

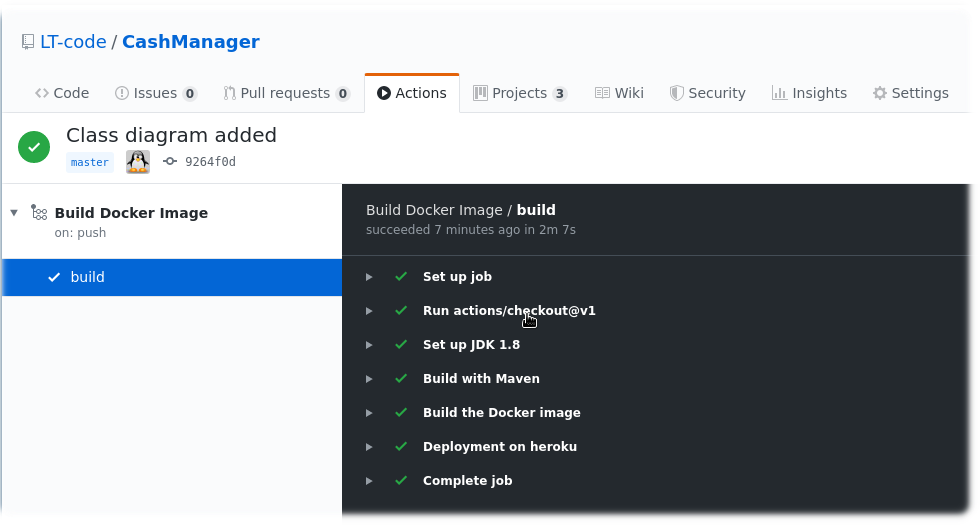


A coverage code of 88.3% for 4920 lines of code.



## Devops :

After each push on the master branch, all tests are executed, the application is exported as .WAR to be moved to a docker where a Tomcat server is located. Github offers an action tab to create piplines that allow this.



Once the image docker build, the piplines send the resutlat on heroku where we can look at the logs of the server.

