

**Internship report**

Programming of a new API in Python (Django) and JavaScript.

RHOULAM Mehdi

Promotion 2022

Epitech Marseille

# Appreciation

It’s never easy for a student to find an internship. So, I would like to thank **Mohamed Bijou,**

who made me discover this opportunity in **Crossdock** to have integrated me into their company.

I would like to thank my internship supervisor, IT project manager, for his patience and for accompanying me throughout my internship.

I would also like to thank **Abdouramane**, an IT intern who finished his master, for giving me a good example of how projects are managed, how to think about a project's architecture,

He also gave me good advice on how to learn fast with good understanding of Django, python and his valuable advice really helped me to improve myself.

Finally, I would like to thank all the employees like **Leila** and **Hakim** the company for their welcomes and their professional seriousness during 6 months of internship.

# Contents

[Appreciation 2](#_Toc15802668)

[Contents 3](#_Toc15802669)

[Introduction 5](#_Toc15802670)

[The company 7](#_Toc15802671)

[Organization Chart 9](#_Toc15802672)

[Context 9](#_Toc15802673)

[Gestion Stock 11](#_Toc15802674)

[Realized projects 14](#_Toc15802675)

[Purchase Orders difference 14](#_Toc15802676)

[Package and products Management 15](#_Toc15802677)

[Smartphone version 19](#_Toc15802678)

[The interface of the site in tablet and smartphone version 20](#_Toc15802679)

[The technologies 21](#_Toc15802680)

[Development Environment 21](#_Toc15802681)

[Data base 21](#_Toc15802682)

[Languages 21](#_Toc15802683)

[Frameworks 22](#_Toc15802684)

[Software architecture 23](#_Toc15802685)

[The MVT 23](#_Toc15802686)

[The MVC 24](#_Toc15802687)

[The MVVM 25](#_Toc15802688)

[Conclusion 26](#_Toc15802689)

# Introduction

To validate my second year at Epitech, each student must complete an internship of 6 months in a company. I was really lucky because I worked at Crossdock as an intern for a period of 6 months from March 04, 2019 to September 1, 2019, located at 122 Allée de la Lavande, 84300 Cavaillon.

During this six-month internship I learned to work in a company and put to good use the knowledge I acquired during my two year at Epitech.

The first two weeks of the internship allowed me to learn new technologies through training under the supervision of Abdouramane **Mahamane**, my internship colleague. The good atmosphere that prevailed in the group gave me the motivation I needed to complete the various tasks that were assigned to me.

The main topic of this course creates from scratch a new API from the help of an older one

In a technology called Filemaker 14 in order to have a consistent long-term support,

And a philosophical envy to have its own api developed by its own employees.

# The company

History

Crossdock was founded in 2013 by **Mr. Mohamed Bijou**.

Crossdock is a recognized logistic provider, ahead in term of « traceability » and « security » for drugstore, in the field of unrestricted medical contents in France they are working for the leader of dietetic groups.

Activities

The main areas of intervention:

* Photograph packaged products at the entrance.
* Delivers products directly.
* Seeks delivery companies for customers.
* Resolves issues between delivery companies and customers or the destinators.
* Change automatic delivery process by the urge for the destinator.

The company has about ten employees at its headquarters in the Vaucluse region and a lot of destinator in France. Each package is checked by a Logistics Agent Dispatcher. The Logistics Agent Dispatcher is the person in charge of verification, each issue must be reported to the chairman.

Organization

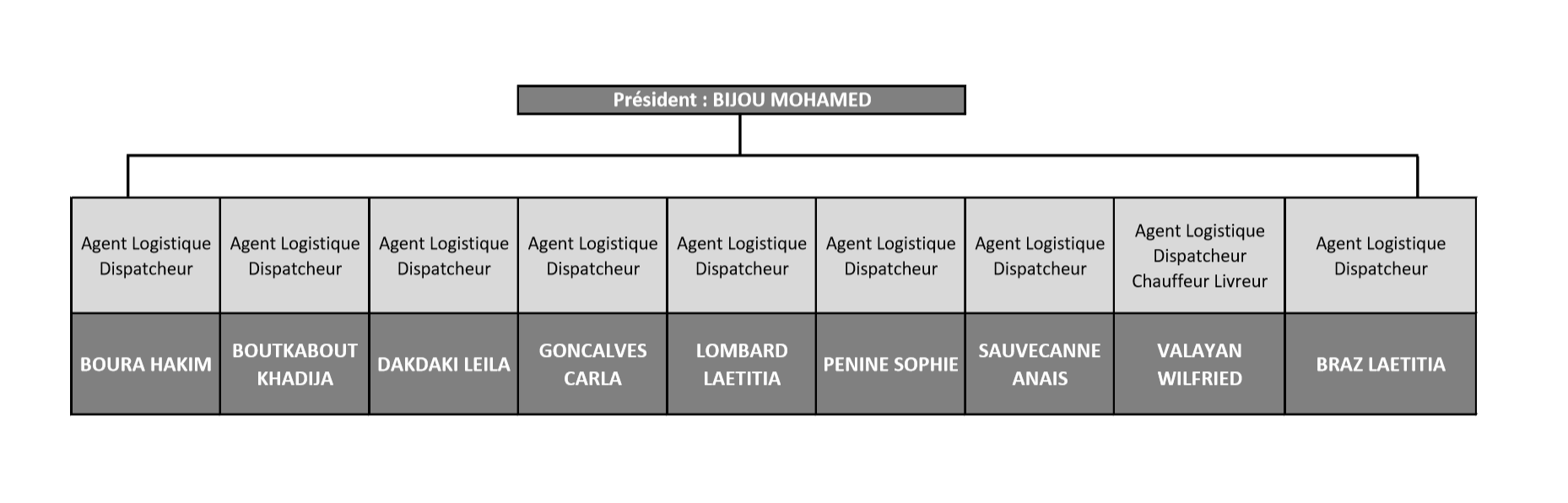
Crossdock has not stopped growing during these last years, in particular thanks to the organization's quality. To respond most effectively to a customer, the company has set up a quality infrastructure. Indeed Crossdock is composed of several crucial points :

* **Supply managing:** Organize the management of the service. Facilitate the different stages of the income/outcome of products from stock. Manage part of the department's files independently for each package received & sent out. Analyze situations and implement elements to solve, improve, innovate. There are different main activities such as setting up and updating supplier or product (article name…) databases, managing the production/non-production supplier panel.
* **Products Managements :** The product are always selected and there has been a report on the benefit it would make to the company and client, each products have a ratio of “risk”, “benefit” it allow the company to have less hard products to treat (usually done by chairman).
* **Litigation / disputes :** With help of Filemaker they can identify each “package”, and select the correct litigation happening, and with helps of “screening” in the zone where we stock, they can prove if it’s their fault or not.
* **Labels / QR Codes :** There is also an usage of QR code, with raspberry pi on python,

The reason to have QR code is not to brag “I have QR CODE !”, not at all, it is a necessity,

In this sector there are a lot of dishonest people, so for the own protection of the company they need to prove what happened during the “logistic” of a product, from the deposit with the delivery, until the last “mile” delivery, they are screened with the help of qr code to identify which packet have been there, if a dishonest or unlucky destinator say he didn’t receive, we can check the qr screens, and see what happened, it can also be useful if the packet have been lost, the delivery company become responsible.

## Organization Chart



# Context

When I came at Cross dock, I had a desktop for me, I was with another internship,

who welcomed me.

from first day I had fixed to myself one job, to begin in a good way my project,

I ignored learning of Django for first week, I worked on FILEMAKER 14,

else I would have no way to understand what was already done, in terms of Database architecture,

of template interaction, of “project” architecture, “why the developer before me has done thing this way”, I had no global “documentation” only some “sentences” before some algorithms, but no context explanation.

After understanding what was done, I copied the architecture of the DB, I learned how Django works, what was the difference between his equivalent cousins, how to work with gets & posts,

when my background with this technology was strong, I began to have a beginning of API,

my first version was really “poor”, but it helped me to have a better database, because each time I do a new try, I see problem in DB architecture logics, so I make it better and I commit each time I do a better job.

When I began the tutor showed me an API in FileMaker, he had to know if we could have an API equivalent or better than Filemaker’s, in FM they had a like 30m of duration for one algorithm of sorting, because FM’s way of doing back-end make it really slow, in python same algo, would have been faster than a 1/10 of FM’s one.

My job was to deliver a working **mock-up,** so another team of developer take it as a “project”,

as an intern I don’t have to do all the “back-end” – “front-end” live “interaction”, it would take me too much time, I suggested to do simple “create” in front “save” with events in back, but the real API should always be in “live” state with help of react, and hot reload with webpack,

this would make this API much more professional, but I couldn’t do all the needed structural work and yet do all these frameworks in 6 months,

I found an issue by doing all the structural work and when I ended, I asked my tutor to let me do one template only in react so we can see deeply the way it can work in future,

but giving a proof that such an API is viable in Python, Django, PostgreSQL as environment

was mainly my purpose for this internship.

# Gestion Stock

During my internship I worked a lot on an application called **Filemaker**. **Filemaker** is a proprietary application made for developing api really fast, it’s made up by Apple inc,

set up by Crossdock to facilitate the management of multiple packages and their traceability as for example being able to modify a package expiration’s date of the application or still consult or modify another type of data site etc…

There are two versions of this:

* **Gestion Stock v1** (Fully programmed with \***Filemaker / sql** Not accessible out of the network through the firewall)
* **Gestion Stock v2** (Fully programmed with \***Django / html / jQuery** Not accessible out of the network through the firewall)

**Gestion Stock v1** will gradually disappear by updating its functionality on **Gestion Stock v2**

We also need to be sure of its sustainability & stability

We are invited to login in order to access the **Gestion Stock** service (user and password are provided by the company’s Chairman)

On **Gestion Stock** the rights to view, modify documents or other rights vary according to the type of user connected. There are 3 types of users to be developed: Admin (Check logs…), employees (do the dispatching stuff) and clients (see what happen to their products).

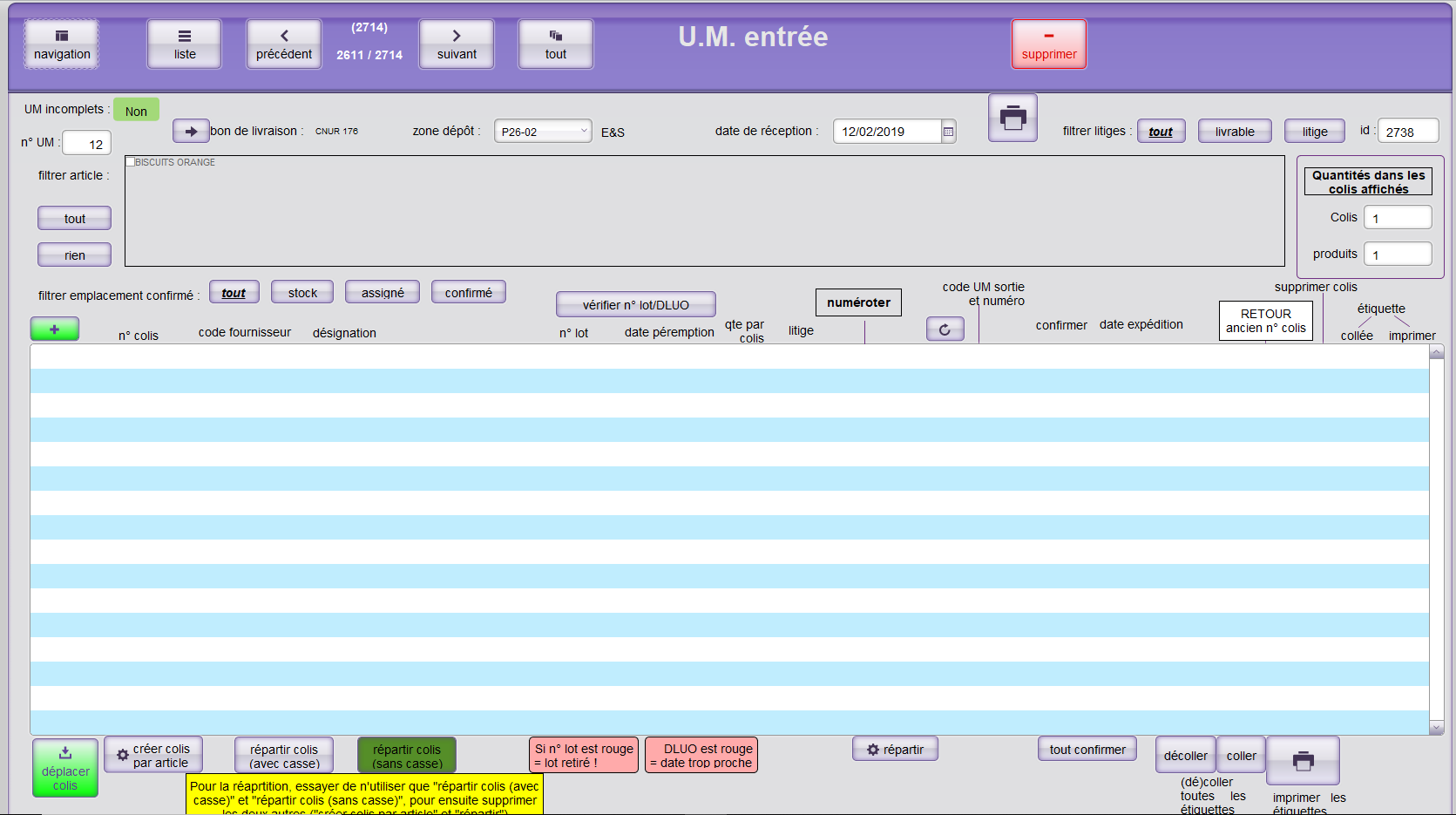
As a developer I had access to **Gestion Stock** with an internal account, which allowed me to have access to all the different **Gestion Stock** functionalities (in a Back-up).

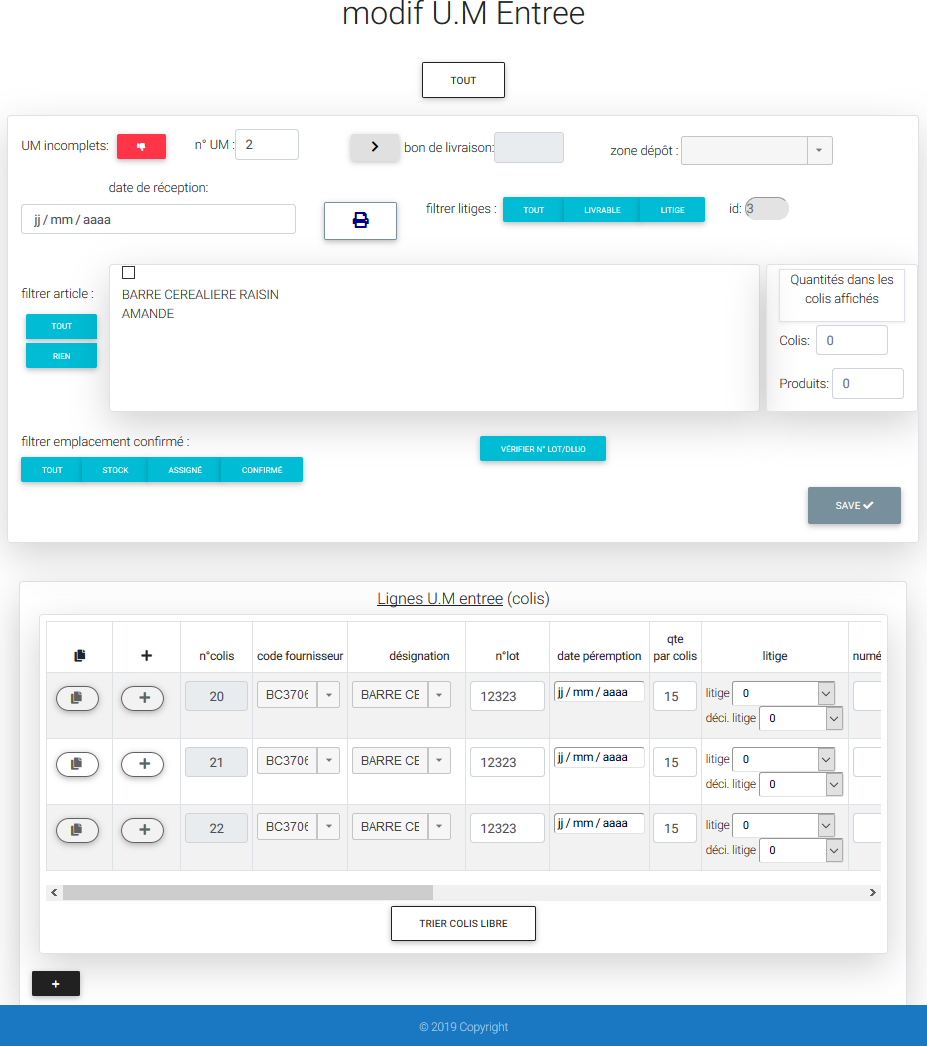
In order to add new functionalities, there is a development process:

* We are developing a functionality on a « **copy** » of the project (with a different database from « **Gestion stock production** » which is the public mode and accessible in local network) This allows us not to have conflicts with « **Gestion stock production** »
* Once everything is set up, we first publish a « **Gestion stock V2 demo »**. This one uses the database of the production version, we can test if everything works well, knowing that the database of the production version is more important in terms of content than the one used in development version.
* And finally, once we have finished testing on the demo version, we will publish on the production version (not yet done, working with a team of devs who will finish this after my internship).

Here is we can see the differents versions of Gestion Stock (In development mode) look like.

**Gestion Stock v1 (Actually used)**



**Gestion Stock v2 (First design of new api)**

# Realized projects

## Purchase Orders difference

5

4

4

4

3

2

1

1. This zone gives us the « id » of which « order » are we in, so we can know what to.
2. This part shows us how many « products » got automatically indexed, by priority…
3. This input allows us to change the « quantity » the machine gave,

if we want to give him more or less than what has been automatically given.

1. This button does a call to the « sorting products », and takes the « gap » between automated data, and manual selected data.
2. This part allows you to create deliveries notes from « sorted » products.

## Package and products Management

As stated, this project needs to have multiple ways to deal with a package,

It can come alone without « notes », like an emergency package for a shop,

Or it can just be a standard one,

And we also have two ways to carry out delivery,

First one is simple, you receive an order, you send stock really simple,

But sometime stock isn’t enough, if you have this issue you’ve got another two issues,

You can send and lose some efficiency, or you pack the stock and wait for a new arrival of stock

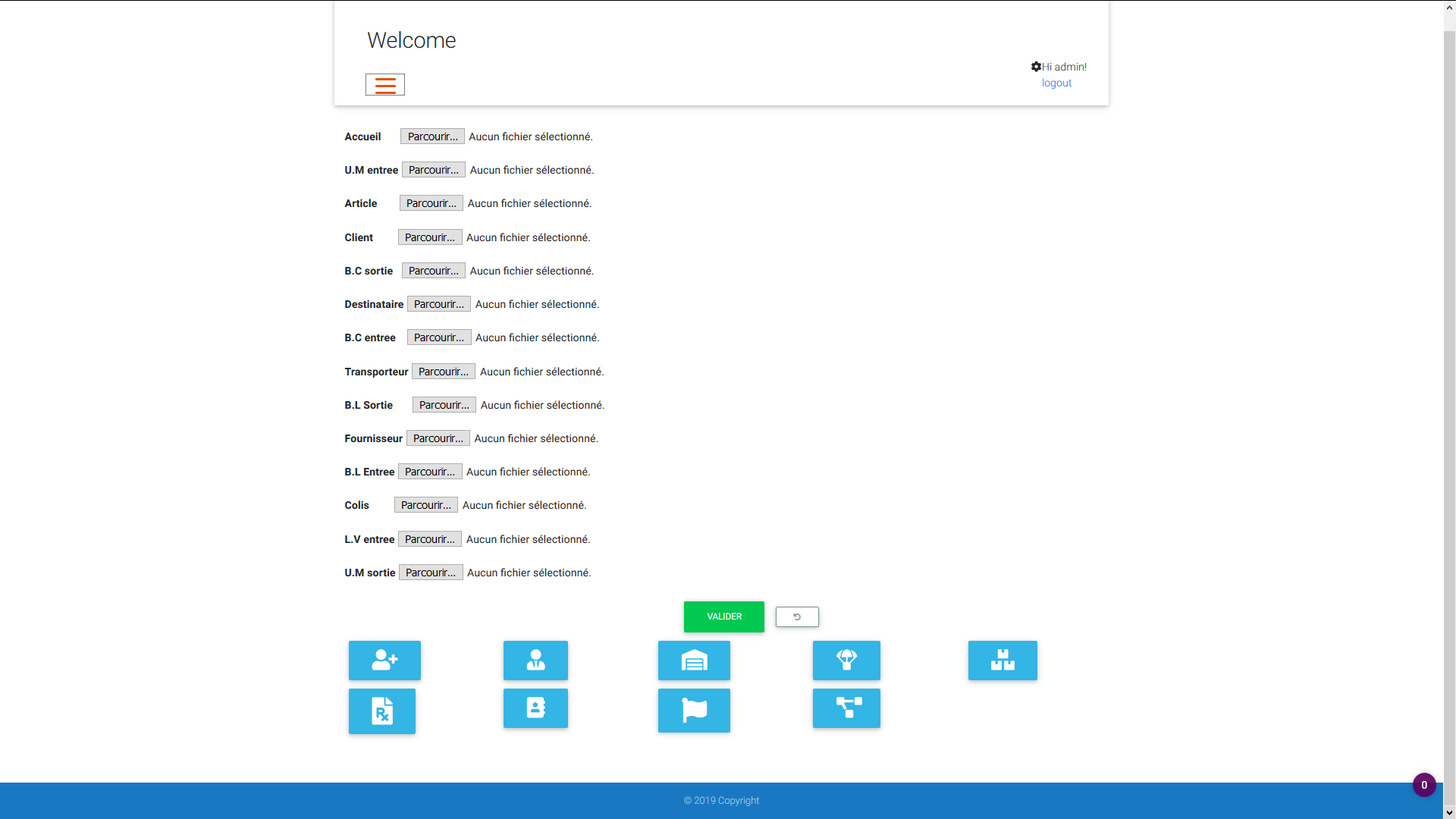
To relaunch the « sort algorithm » and when all orders are respected, we close the pack

The project was carried out in multiple part:

1. **Administrations**: This part can only be seen and configured by administrator,

It has for purpose to create, delete, or modify the needed data,

Like « client's name », his « id » or where he « stock » and so on…



As we can see on the image above, I can change all data that makes the project « runnable »

In this part of the project we can change « images » of menus,

Create, delete users, or simply list them,

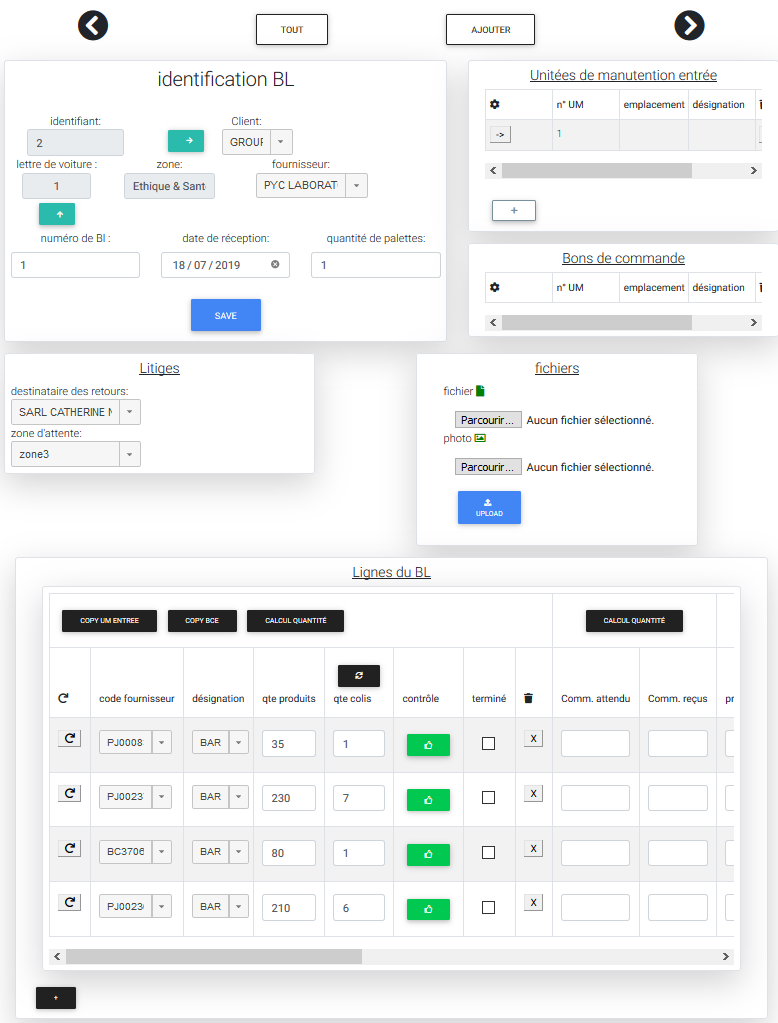
I can also work on what a client can have,

this is the zone where we can deeply manage it,

I can add some identification for a zone for clients, same for suppliers, destinators, articles etc.

And in generals we can manage what « type » of role a contact (is he assistant?) can have,

The countries available for deliveries, or what type of « issues » we can have on a delivery…



1. This button checks all the “Unite Manutention entrée” and write a synthetization of the stock in this template.
2. These select combobox from jQuery can do an auto complete from the DB, with same code and designation for a same article.
3. This button call an event who reset the two select “options” cause when you select one of them it clear the option of the other select, so the employees can’t make a mistake, this button allow to reset the select in case of a miss click.

3

1

4

2

This configuration is important for the second part. Indeed, it will make it possible to see package not properly stocked in the good “zone” in the warehouse, and we can search if we have traceability if a package needs to be destroyed for many reasons, like a risk of toxicity.

1. **Orders from destinators:** This part is visible and modifiable by the interveners. It allows you to call for products, with a priority parameter, we receive it from the “main” client,  
   its purpose is to get information on who need what, without this data, no sorting of algorithm  
   can be possible, with a CSV (Comma separated Values), we can’t know what to do,  
   we should do all the information for “pre-sorting” manually, this CSV allow us to have an automation, orders creation, and with the helps of orders “information” we can sort products to each orders with the respect of priority, date of lapsing, and if the batch number is still allowed for delivery.



On this illustration we can see the CSV (had to blur it for administrative reason):

* **Left:** In this csv we have 7 column of data for one entry, first one is the id of an order, after this the second one is the name of the client ordering, third is the destinator the client aim for receiving his order, forth is the code used to “identify” this destinator (two destinator can have same name but different code of identification), fifth is the transport company used by the client, sixth is the number of command usually same as first column but without “letter”,  
  seventh is the “date of command” it help us to create an order with a “creation” date of this value but one problem is the standard used is using slash instead of a hyphen, so we have to translate it in standard html format (YYYY-MM-DD in back-end value) yet it show YYYY/MM/DD in template, now last value in this csv is called “source” it is the “unique” name identification of a client, if we have more than one client with same name this “source” make us able to identify which one of both is the client we seek.
* **Right**: In this csv we also have 7 column, it help for error checking, the first value is again an id to identify the line of order, second value is the “foreign-key” of the “order” received in the first “csv” we aim for the first value of the left csv, who have created an order, if one value is wrong we cancel everything, the foreign-key of order is there to know which “line” of order it is (they are the way we have to know what is going in an “order” received in left csv),

We have after this the Article code to identify which article is ordered, we got also a safe value in case there is a problem, it is the name of article, finally we have de quantity ordered, and for security reason the “source” to ensure the good client is ordering.

It should be noted that this part is still in progress and therefore the presentation is not complete. There will be additional features. A mobile version is planned for data checking at weekends or days when employees can be out of work,

If a client say he have an issue with a package we will be able to trace where he was,

If he was sended, and thanks to raspberry we could say your packet got packed, “we have a picture who proves it”, this is the main purpose for the next development of a phone API, but yet not planned.

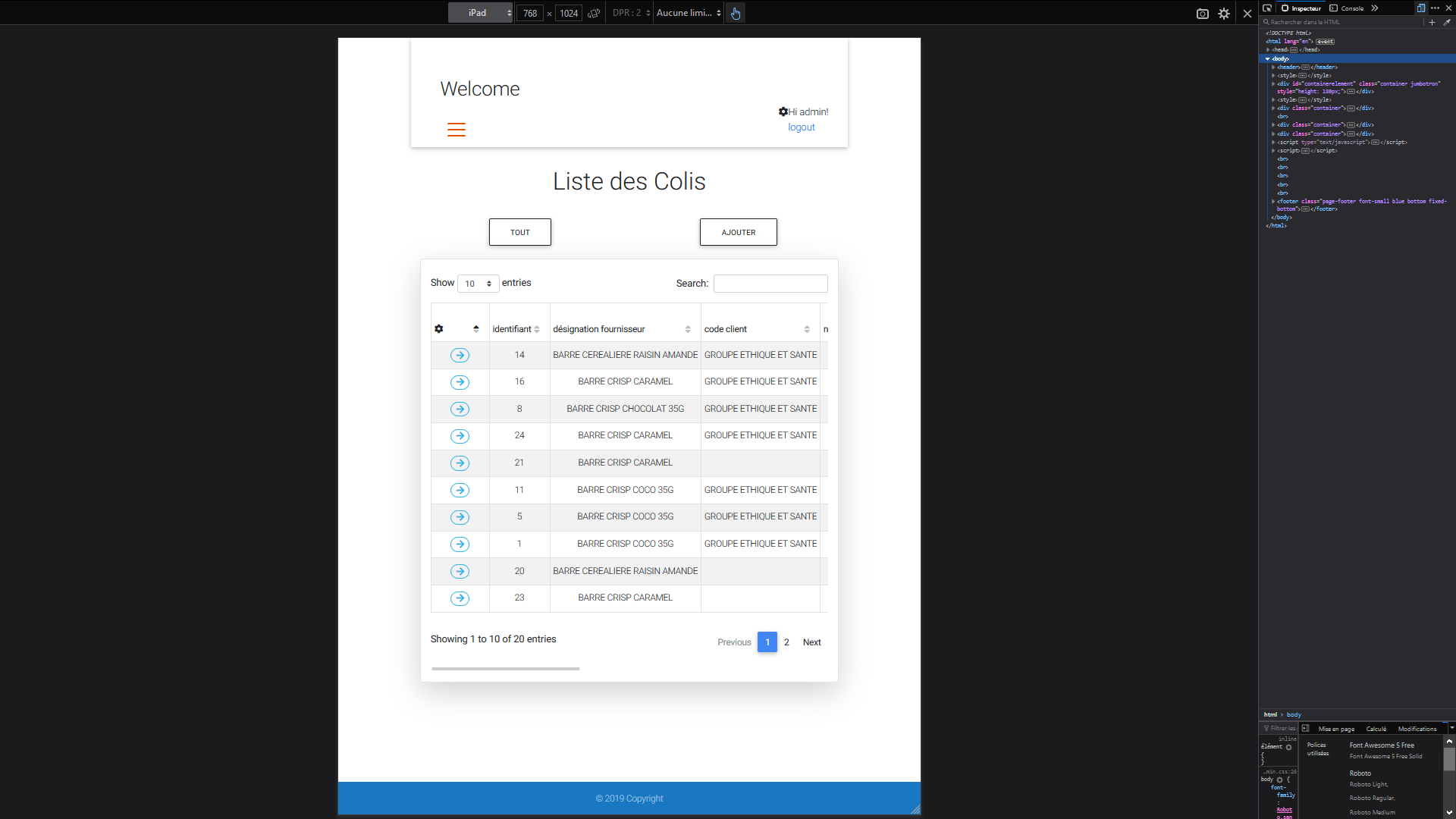
# Smartphone version

The **Gestion Stock v1** site was not designed for use on smartphones or tablets, which is very

impractical when you don't have a computer with you.

**Gestion Stock v2** uses a Bootstrap. Therefore, it is possible to make it compatible with the smartphones thanks to the mobile first design used by Bootstrap.

## The interface of the site in tablet and smartphone version



Tablet

**IPhone Xs**

The presentation of the site doesn’t really vary according to the size of the screen of the support used. On a larger interface we have a menu which is displayed bigger.

While on an interface smaller you have to scroll on menu button to display the choices.

We designed the app to look same on phone, tablet and pc, by using simple « table » logics,

And tried to have the less interaction between api and users, most data are found by the « views »

User only have to select good value and « use » it most of the time.

# The technologies

## Development Environment

During the internship I was working under Windows on IntelliJ which is an editor for Java(good for javascript), HTML and Javascript editing by JetBrains.

JetBrains has several advantages that facilitate the use of the software such as :

* **Major frameworks supported**: IntelliJ is perfect for working with Django, WordPress, Laravel, Magento, CakePHP, codeigniter, and other frameworks.
* **All Python tools**: The editor actually 'gets' your code and deeply understands its structure, supporting all Python language features for modern and legacy projects. It provides the best code completion, refactorings, on-the-fly error prevention, and more (we got 3 lvl of Highlighting, None, Syntax, Inspections).
* **Intelligent Coding Assistance** : Hundreds of inspections take care of verifying your code as you type, analyzing the whole project. It support external documentation, code (re)arranger and formatter, quick-fixes, and other features help you write neat code that’s easy to maintain & the most important auto-indent for python hard laws on indents.
* **Fast and Safe Refactoring** : Refactor your code reliably with safe Rename, Move, Delete, Extract Method, Inline Variable, Push members Up / Pull members Down, Change Signature, and many other refactorings. Language-specific refactorings help you perform project-wide changes in a matter of a few clicks, and can​ be safely undone.

## Data base

**PostgreSQL** (PSQL) is a relational and object database management system administration and development. We use it for database management.

It suits the need of scalability needed in this sector of logistic

## Languages

There are two types of language used for our application:

* Client-side language (JavaScript, front-end) allows to manage the user interface: display, interactions (buttons, etc).
* Server-side language (Python / Django) allows to process a request sent on the client side and returns a response in JSON format or do some redirection with some GET request in URL.

## Frameworks

During this internship I discovered many Frameworks:

* Django that can be used to develop web applications and services using Python2 or 3, and provides 100% object-oriented code using a broad spectrum of language features,

Django helps you build data-intensive, cross-platform web apps for desktops, tablets, and smartphones. Take advantage of powerful features in with PostgreSQL who can be really good at scaling data-size,

With the help of python, we can do almost anything we want, with ease.

* React Framework (Front-end), not yet done, but I have begun to read books, and tutorial on this framework, it will consist to make my front-end more “read-able” for humans (developers), by using “modules” instead of doing only “files” where I put function, these modules will be for each “interactions” I have in my front, it also allow me to use “BABEL” for JavaScript,

BABELjs allow a good interaction between “old” browser, and new gen available, I just have to write modern code, and it will translate it on a way IE7 can understand, also if the function I use isn’t implemented in the runtime engine of IE7, it is really good because if I use latest function who makes my front end 2x faster than it would be on last update of JavaScript, it could still work on IE7, it makes you win a lot of time, cause you don’t lose time on interoperability.

React, is one of the frameworks I used with the help of another framework for backend compatibility with Django (Django Rest Framework),

It will give me an ability to have a better “front-end”, with less “laggy” events.

* Django REST Framework, is a good toolkit to build an API,

It can make using a front-end like react, really easy, instead of

Doing a back-end different because of react, we just have to use

It to make “Django” in harmony with React.

* JQuery, when I needed good widgets, and fast and reliable components,

jQuery was there for me, making JavaScript events easy to use,

jQuery is easy to use, easy to understand, and easy to maintain,

this is the reasons it still exists trough time, also if there are a lot of competitor,

and new front-end framework more “complete”, jQuery got one big advantage,

you just need 1-3 ~line of code to make it work, the syntax is easier than the one JavaScript uses, so these are the reasons why jQuery is still strong in the market,

but it is mostly until I’m able to use react (it will be done for one template, all the others will stay on jQuery).

# Software architecture

## The MVT

MVT is a design patterns, mostly used to organize our code files, it follows the logic of mvc.

**Model**: Its where we create the « Database » structure,

Django can recreate database structure from models objects declarations,

We can also select which data we send by default on object call,

We can decide of foreign-key deletion behavior.

Django’s orm allow also to do “function” in models, for doing calculation or any algorithms without doing it in views or template.

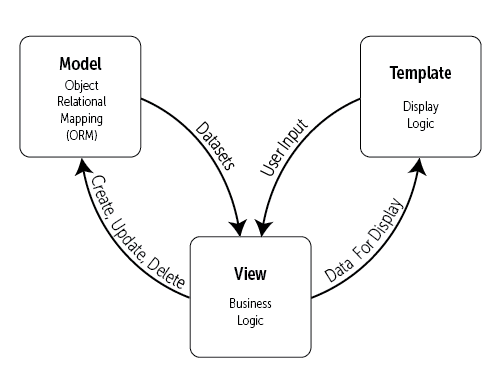
**View**: Part that select which data is sent to a template, or the behavior to have on the request sent by template to him,

He can do whatever he wants on model, he is the « middleman » of Django.

The way Django differs from its cousins, is that he can do the job in python,

It makes it easier to understand, and we can use python for a lot of operation and be faster than php equivalent

**Template**: This part manages the logic of the code that takes decisions. In a way, it is the intermediary between the model and the view: the view will ask the model for data, analyze them, make decisions and return the text (data) to be displayed to view. The view contains exclusively Python. it determines whether the visitor has the right to see the page or not (access rights management) with Django user management system (we can override it to have it suited for the project we need).

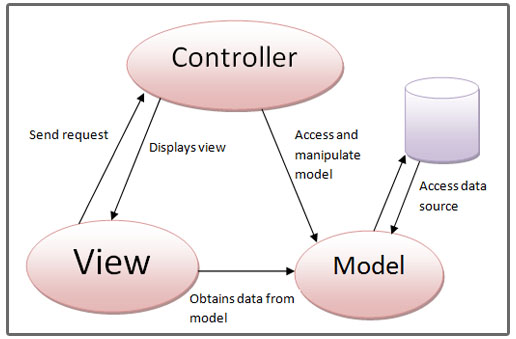


## The MVC

**Model**: Element that contains the data as well as logic related to the data: validation, reading and recording. It can, in its simplest form, contain only a single value, or a more complex data structure. The model represents the universe in which the application is embedded. For example, for a bank application, the model represents accounts, customers, and transactions such as deposits and withdrawals, and checks that withdrawals do not exceed the credit limit.

**View**: Visible part of a graphical interface. The view uses the model, and can be a diagram, a form, buttons, etc... A view contains visual elements as well as the logic needed to display data from the model. In a typical desktop application, the view obtains the data needed to present the model by asking questions. It can also update the template by sending appropriate messages. In a web application a view contains HTML tags

**Controller**: This part manages the logic of the code that takes decisions. In a way, it is the intermediary between the model and the view: the controller will ask the model for data, analyze them, make decisions and return the text to be displayed to view. The controller contains exclusively PHP. In particular, it determines whether the visitor has the right to see the page or not (access rights management).



## The MVVM

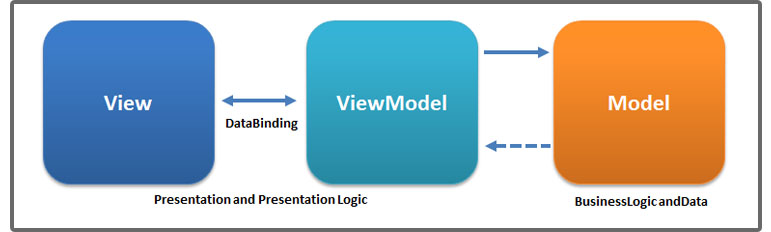
The MVVM model has been specially designed to improve the separation between the data and the view that displays it. The link between the view and the data is done by binding mechanisms.

The binding is a mechanism which makes it possible to make links between data dynamically. Which means that if A and B are related, the fact to change A will be passed on to B and vice versa.

**Model**: As for the MVC, the model contains data. Usually, this data comes from a database or an external service such as an API.

**View**: As for the MVC, the view corresponds to what is displayed. The view contains the different graphic components (buttons, links, lists) as well as the text.

**View-Model**: this component makes the link between the model and the view. It manages data links and possible conversions. This is where binding comes in.



# Conclusion

First of all, I would like to thank Crossdock again for giving me a huge chance to join them.

I had the chance to have a really good project for my second year,

Working on an api from scratch on a framework I didn’t even knew, nor the knowledge of Python language, and same for Filemaker, I also had to learn its syntax.

I had to understand the way Filemaker manage his own database, the way it has to “stock” each foreign key data in his own table (each view has her own table), because it can’t live access data in his “views”

I still have 1 Month in my internship at the moment, what I was asked to do is a « ready to use » API of course it cannot be put in production before to have it tested a lot and change the way I create instance in “front-end”, I have to create temporary object and save or modify in “back-end” to be sure nothing can break,

I will sustain what I did by making it safer to have it riskless. It can therefore be said that the goal set for me at the hiring process, has been achieved!

This internship interested me a lot,

When I was seeking an internship, I had some “strong” basis on “technical project”, so I was in research of a challenge that could keep me in the “mood” of working hard to achieve it.

I didn’t want to be in a “css-bootstrap” only project, I was going only if I could have a true “project" who ask from me to be at 200% each day, to have a working and pretty API,

It asked me to do a lot of UX/UI design, with the respect of a rule I learned, a better looking design can be a “harder” coding experience, I had to cancel some design pattern & UI I wanted, I took a step back did easier design, make it work and when it was working I was trying to make it better looking,

In some terms I can say it allowed me to deepen my knowledge in computer science and to acquire additional experience in the field of “web” API development,

It would be a pleasure to work again in this “sector” with such a good company that let me do what I think is the best, and to show and give respect to my work, also if it can be pretty “poor” at the start, but with confidence earned each month, and trustful tutor, you can only become better!