Prototype Acceptance Plan
For
Tag&Try

BOUBRIK Charazed

CAO Ying-Xi

ELBAZ David

LETENDART Pierre-Alexandre

PARTOUCHE Natanel

SIDI KHOUYA El Mahdi

ING5

ECE PARIS
ÉCOLE D'INGÉNIEURS

PAP- PFE13-59-20140205

Acknowledgments

The authors are very much indebted to ECE for having provided the environment where the study has been developed, to Frank Bietrix, cluster responsible for his guidance during the project.

We would also like to express our gratitude to Virginie Corvellec, responsible of valorization start up for her help in entrepreneurship.

We would like to sincerely thank our mentor, David Shapira for his invaluable guidance and support at each step of our project.
Their guidance and suggestions have been very valuable.

Moreover, we would particularly like to thank M.Waller for his recommendations and all his help in marketing and in communication.

In addition, all the team work would like to acknowledge the support provided by José, CTO NetDevice.

History Version

Date	Version	Type of changes
01/13/14	v1.0	Creation of the template
01/16 /14	v1.1	Beginning of writing the PAP
01/ 30/14	v2.0	Finish the report

Overview

This document presents the Prototype Acceptance Plan (PAP) of the project Tag&Try. The project Tag&Try is a new concept store: this is an application that may do shopping easily in shops and customers can share their feelings about this new experience of social shopping. Firstly, we introduce the context and the objectives of the project in introduction. Secondly, we present the stakeholders and the project management. Then, the working methodology will be explained.

Table of contents

1 Introduction

1.1 Context

The E-commerce grows up very fast in France and everywhere in the world. It's taking more and more market share besides the shopping in real stores. Mobile commerce is merging E-commerce and Real store. The Tag&Try mobile application is coming to help the store to improve user experience in Store. By making connection between the Real and E-commerce, we can improve the shopping routine and use some E-Commerce concept in the real commerce. NFC & Barcode Mobile service for retailers reduce queue in store. Physical stores are face to: long queue at the fitting room, large amount of cloth in shop take room and make easy messy in stores, waitress folds cloth and find size beside of selling. Moreover, the no data are available: we do not know if the effective merchandising works on the customer, the owner of the shop never know what the shopper behaviour in store is. Also, when the user leaves to store, the store has no possibility to keep contact with the user.

1.2 Solution

Tag&Try is a Solution for retailers to reduce waiting time in shop on fitting room and cash register. Transform the regular shop in showroom and allow to user to select article, ask for their size and colour and add items to the try list. At the end of the order, the users try the cloth, pay them and leave. Smart Queue algorithm, Cross-Selling, Up-Selling engine are the key value of the product. Create a personal cart for the user that will help him to use select items and book a fitting room. The application is a solution that learns about the user behaviour in store. Personalize campaign using the user selection in store. The store learns to know the tastes and trends of user optimize merchandising and waiting cab and box time. Tag & Try incorporates an engine cross selling and smart retargeting.

1.3 Objectives

The first objective of the project is to realize a prototype of a new shopping experience in store. We want to create interactions between user in store, seller and products. The main goal is to figure out if the workflow we are proposing for the store is validated. So, we want to deploy it for various types of shop and several brands.

2 Presentation of stakeholders

2.1 The Team work

Surname	Firstname	Email	Major	Role
BOUBRIK	Charazed	boubrik@ece.fr	Financial Engineering	Algorithms
ELBAZ	David	elbaz@ece.fr	Financial Engineering	Application Development
CAO	Ying-Xi	cao@ece.fr	Telecoms &Networks	Server
LETENDART	Pierre-Alexandre	letenden@ece.fr	Financial Engineering	Administrator Interface
PARTOUCHE	Natanel	partouc@ece.fr	Information Systems	Project Manager
SIDI KHOUYA	El Mahdi	sidikhouya@ece.fr	Embedded Systems	Electronics Development

2.2 Skills mobilized for the project

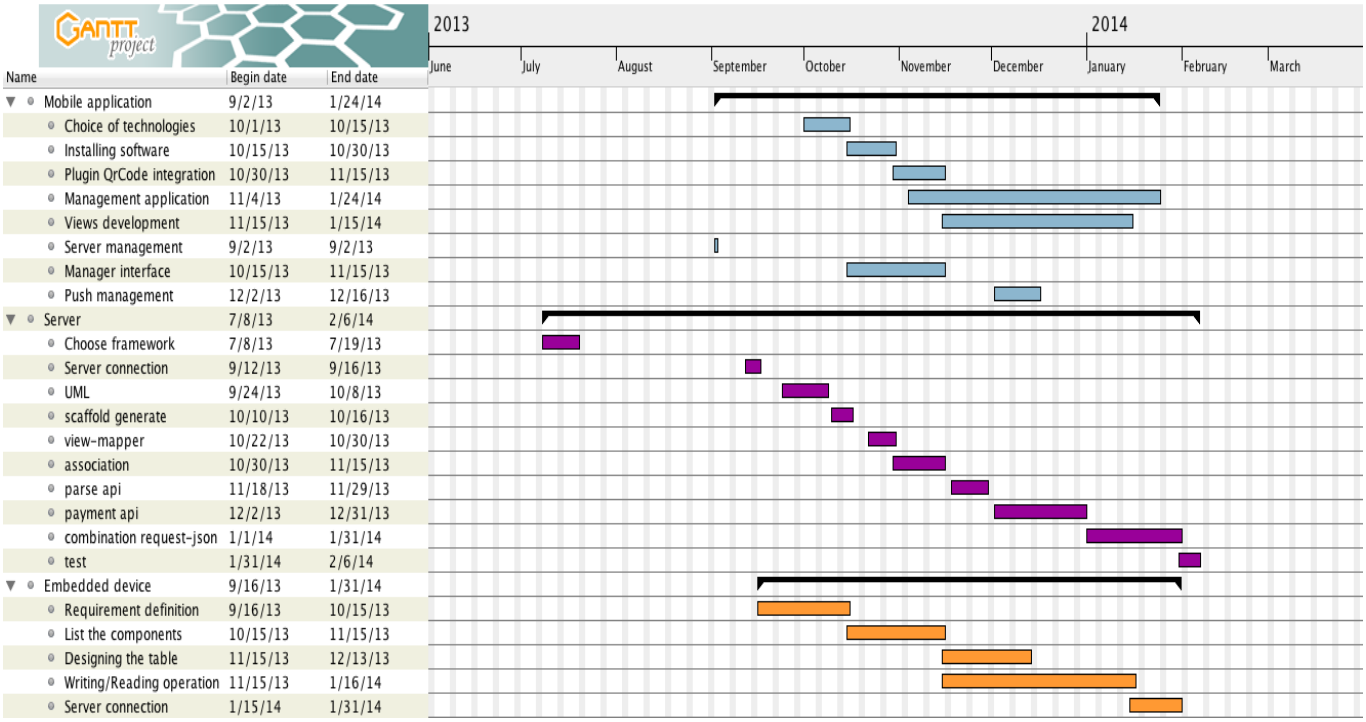
Full Name	Skills	Competences
Natanel PARTOUCHE	Startup Team Management Business Social Marketing	IOS Android WebServices
Ying-Xi CAO	International Background Asia Marketing MVC Framework	WebServices: Ruby on Rails, Js
David ELBAZ	Team Management Finance	Android, CrossPlatform HTML, CSS, Javascript
Charazed BOUBRIK	Artificial Intelligence Finance Marketing	SQL Data Base
El Mahdi SIDI KHOUYA	Hardware developement Firmware Developement	Hardware : Proteus, ISIS Firmware : C, Java, HTML
Pierre- Alexandre LETENDART	Business plan Interface developement	Finance HTML5, CSS

2.3 Other actors

Surname	Firstname	Email	Role
BIETRIX	Frank	bietrix@ece.fr	Responsible Cluster « Communication & Systems »
CORVELLEC	Virginie	vcorvellec@mandarinecodi.com	Mentor Valorization « Start Up »
SCHAPIRA	David	schapira@ece.fr	Mentor

3 Project Management

3.1 Planning



Name	Begin date	End date
Mobile application	9/2/13	1/24/14
Choice of technologies	10/1/13	10/15/13
Installing software	10/15/13	10/30/13
Plugin QrCode integration	10/30/13	11/15/13
Management application	11/4/13	1/24/14
Views development	11/15/13	1/15/14
Server management	9/2/13	9/2/13
Manager interface	10/15/13	11/15/13
Push management	12/2/13	12/16/13
Server	7/8/13	2/6/14
Choose framework	7/8/13	7/19/13
Server connection	9/12/13	9/16/13
UML	9/24/13	10/8/13
scaffold generate	10/10/13	10/16/13
view-mapper	10/22/13	10/30/13
association	10/30/13	11/15/13
parse api	11/18/13	11/29/13
payment api	12/2/13	12/31/13
combination request-json	1/1/14	1/31/14
test	1/31/14	2/6/14
Embedded device	9/16/13	1/31/14
Requirement definition	9/16/13	10/15/13
List the components	10/15/13	11/15/13
Designing the table	11/15/13	12/13/13
Writing/Reading operation	11/15/13	1/16/14
Server connection	1/15/14	1/31/14

3.2 Project management tools

3.2.1 Versioning

Versioning is a software version management

It's a svn system that may manage works of each member of the team work. So, it is a very good tool for working efficiently and easily.

3.2.2 Azendoo

During the project, we used Azendoo as a teamwork tool. It may organize your tasks, plan your projects, share your documents and sync with your team, without email. Azendoo is a web based collaborative task management software. Thanks to this tool, we can communicate between co-workers and exchange information about the project.

4 Working methodolology

4.1 Technical choices

4.1.1 Cordova : Open source

The mobile application had been developed with an embedded view in a native application: Cordova. We used HTML & JavaScript languages to deal with view in the Cordova application. We choose to develop the mobile application using a cross platform technology. We really need not precise interaction with the hardware of the mobile. Cross Platform technologies with allow us to have a fully functional prototype for IOS, Android & Windows Phone. Apache Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript.

<table>
<tr><td colspan="2">View (Display, Actions)</td></tr>
<tr><td colspan="2">AnguleUI</td></tr>
<tr><td>AnguleJS Framework</td><td>AnguleJS Services</td></tr>
<tr><td colspan="2">PhoneGap(Cordova)</td></tr>
</table>

4.1.2 AngularJs : Framework MVC

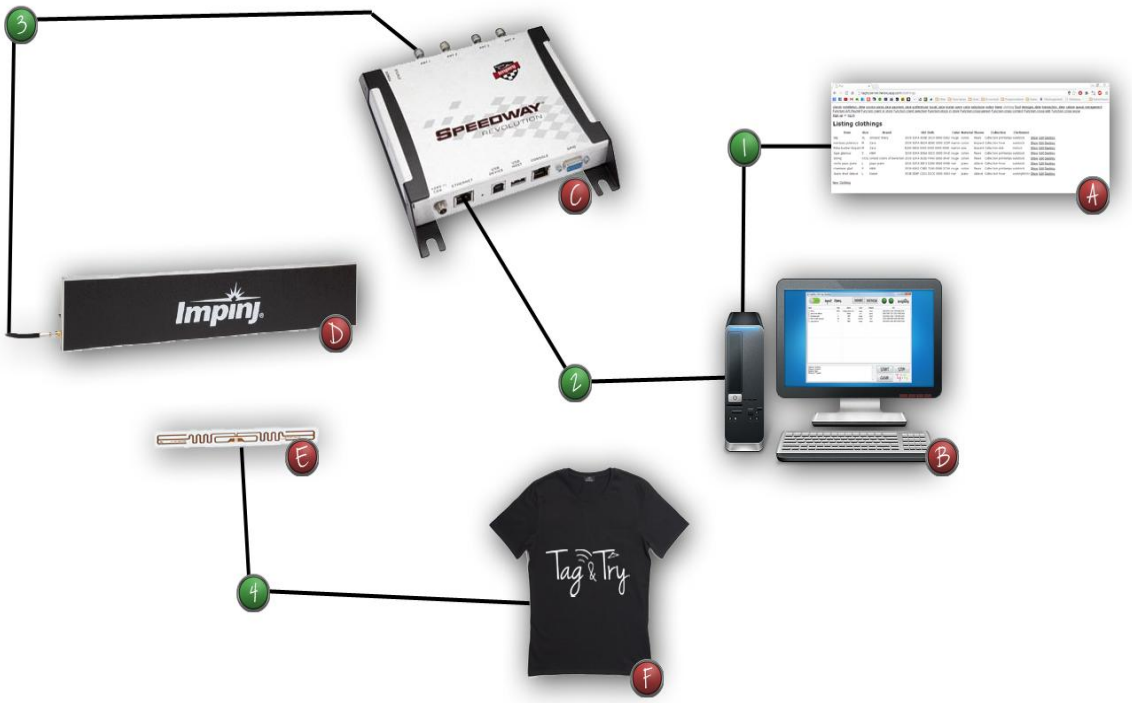
We worked with AngularJs as a framework to manipulate view as MVC (Model View Controller) architecture. We created associated services to this basic model in order to grab data from web service. AngularJs allows us to declare some url routes to redirect the right views.
AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single page applications. Its goal is to boost browser-based applications with model View Controller (MVC) capability, in an effort to make both development and testing easier. AngularJs is a fully integrated and optimized framework for mobile applications.

4.1.3 Ruby on Rails : Web service

We worked with Ruby On Rails (ROR) in order to develop the Web service (API, Interface Admin). It is an open source web application framework which runs on the Ruby programming language. It is a full-stack framework: it allows creating pages and applications that gather information from the web server, talk to or query the database, and render templates out of the box. As a result, Rails features a routing system that is independent of the web server.Like many web frameworks, ROR uses the Model View Controller (MVC) pattern to organize application programming. Ruby on Rails emphasizes the use of well-known software engineering patterns and principles, such as active record pattern, Convention Over Configuration (COC), Don't Repeat Yourself (DRY), and model view controller (MVC).

4.1.4 RFID Handling

In order to supervise the input/output between the stock and the fitting room, we chose an RFID handling by using an IMPINJ Development kit.



A - the server : tagtryserver.herokuapp.com

B - the software explained later

C - the RFID Reader : Impinj Speedway Revolution R420

D - RFID Antenna : Impinj Threshold Antenna A031 1EU1

E - RFID Tag

F - An item from the shop.

1 - the software is sending HTTP requests to the server to refresh the database, we use the URL : "http://tagtryserver.herokuapp.com/clothings.json"

which returns a list of objects that are on the database.

and we use the request :

http://tagtryserver.herokuapp.com/choose/outofstock/3.json

for the tags on the output list. and the request :

.http://tagtryserver.herokuapp.com/choose/backtostock/3.json

for the items in the input list.

2 - the RFID Reader is connected to software with an Ethernet connection

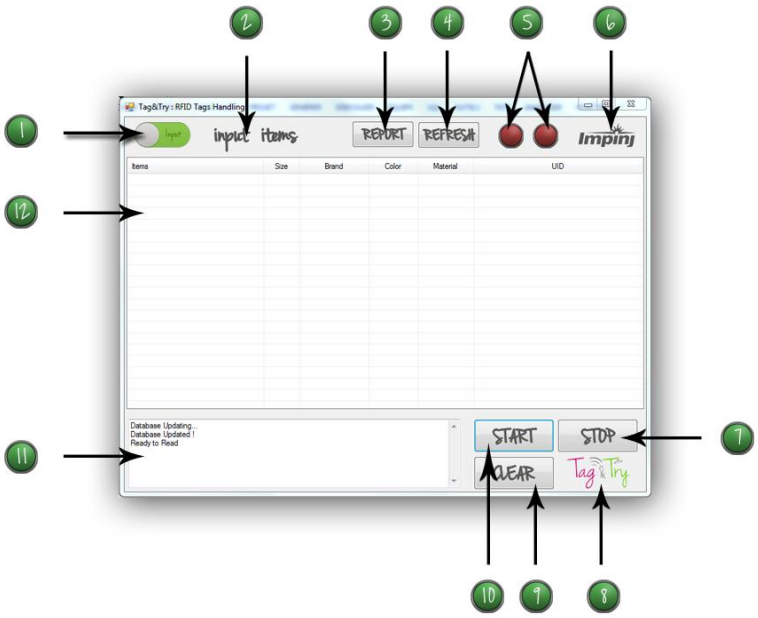
using the default IP adress of the reader : "169.254.123.18"

3 - the anttenna is linked to the reader to detect any tag

4 - Each item from the stock is equipped with an RFID Tag

4.1.5 The C# Software

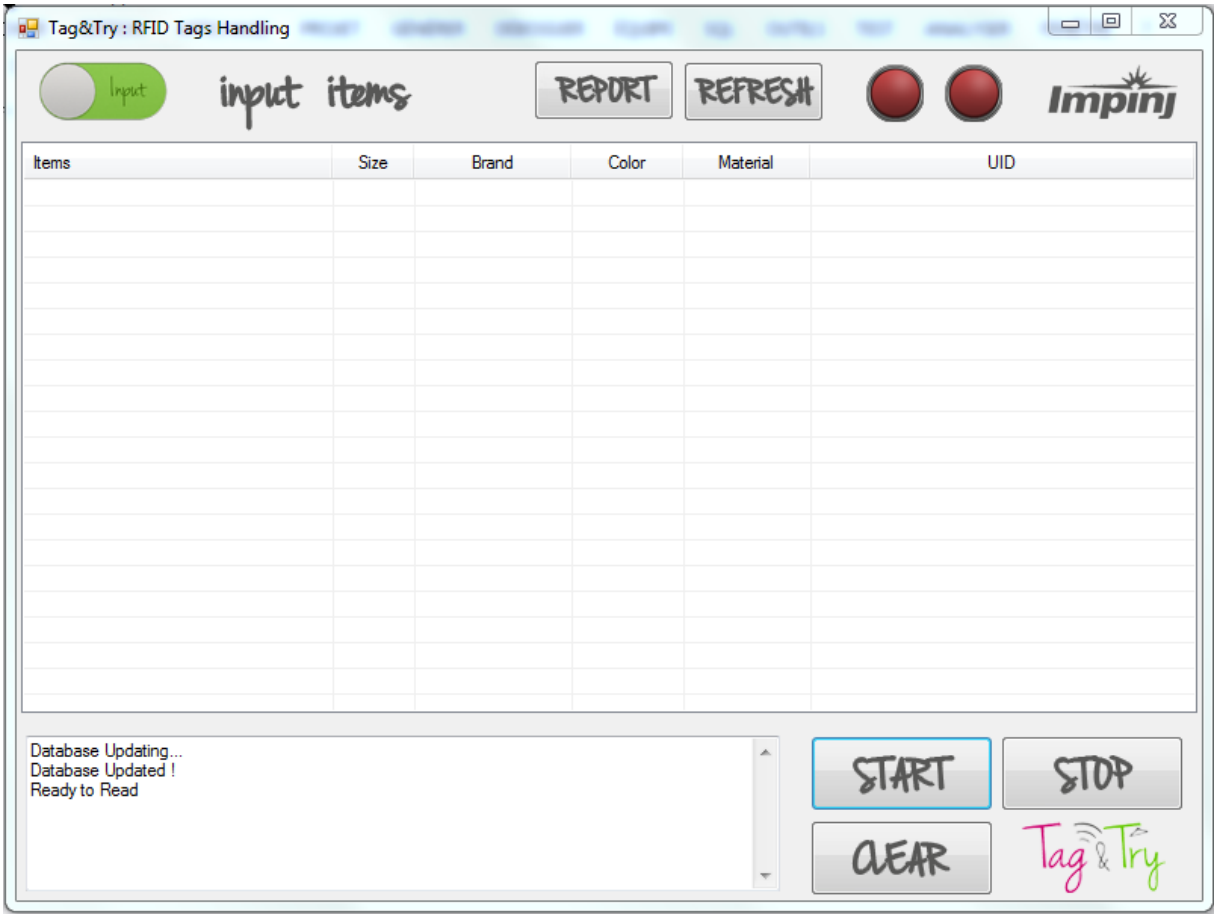
The HMI is composed as follow:



1- a Button to switch between the Input/output modes. The input items are the items going from the fitting room to the stock. And the Output items are the items going from the sock to the fitting room.
2- a label showing the state : Input Items or Output items
3 - a button to display the state of the system in the console (11): the number of items in the input lis, the output list, and in the database. And it also show the date of the last database update.
4 - a button to update the database at any time.
5- 2 Leds to show the state of the reding. Red when the reading is stopped, Green when the reading is in progress and Orange when there is a problem whith the system.
6- a logo of the Reader manufacturer that we are working with : Impinj
7- a stop button to stop the reading and disconnect the reader.
8 - a logo of our company : Tag&try
9- a clear button to empty the console (11).
10 - a start button, to start the reading
11- the console to display any action happening in the system : Start, Stop, Errors,...
12 - a table showing the items read : the title of the item, the size, the brand, the color, the material, the UID.

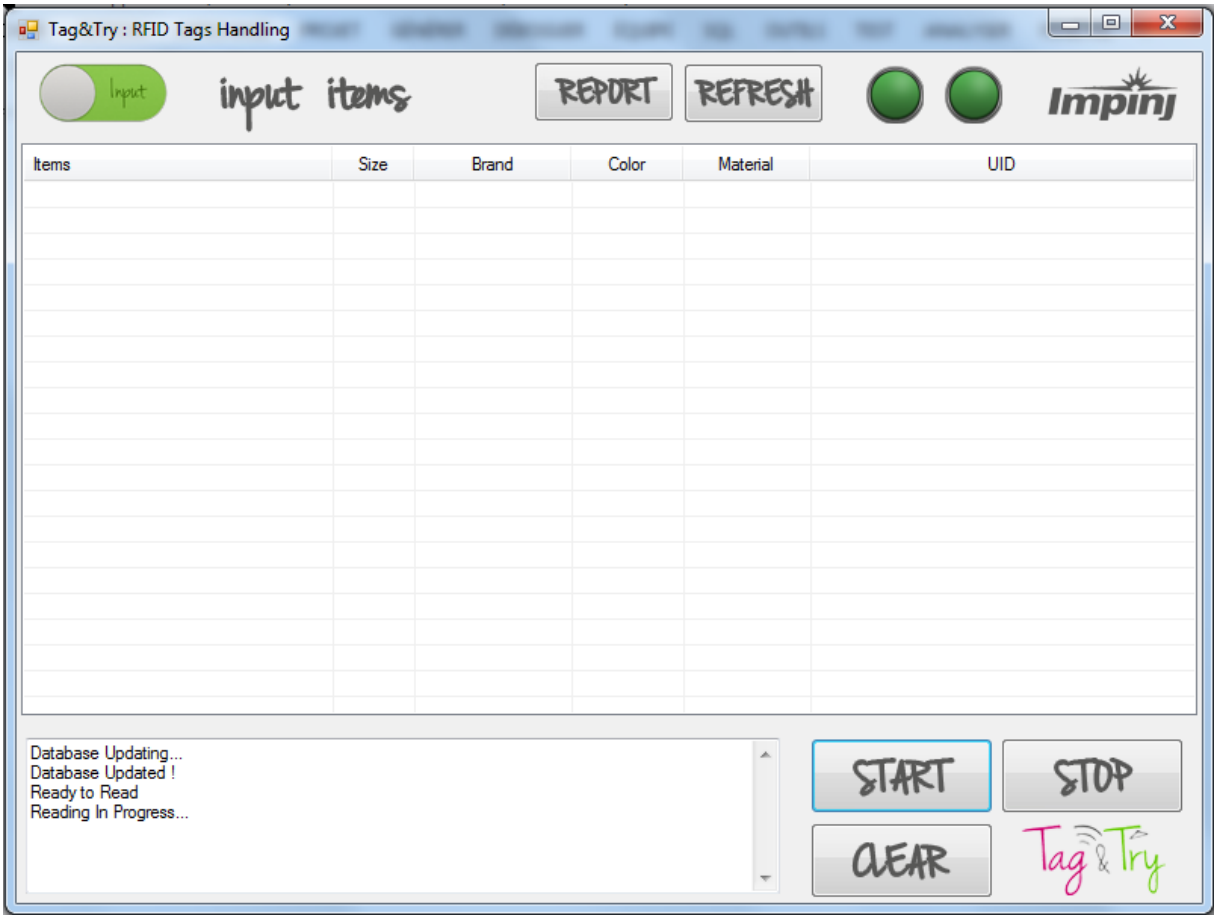
The working flow :

The software start with this HIM :

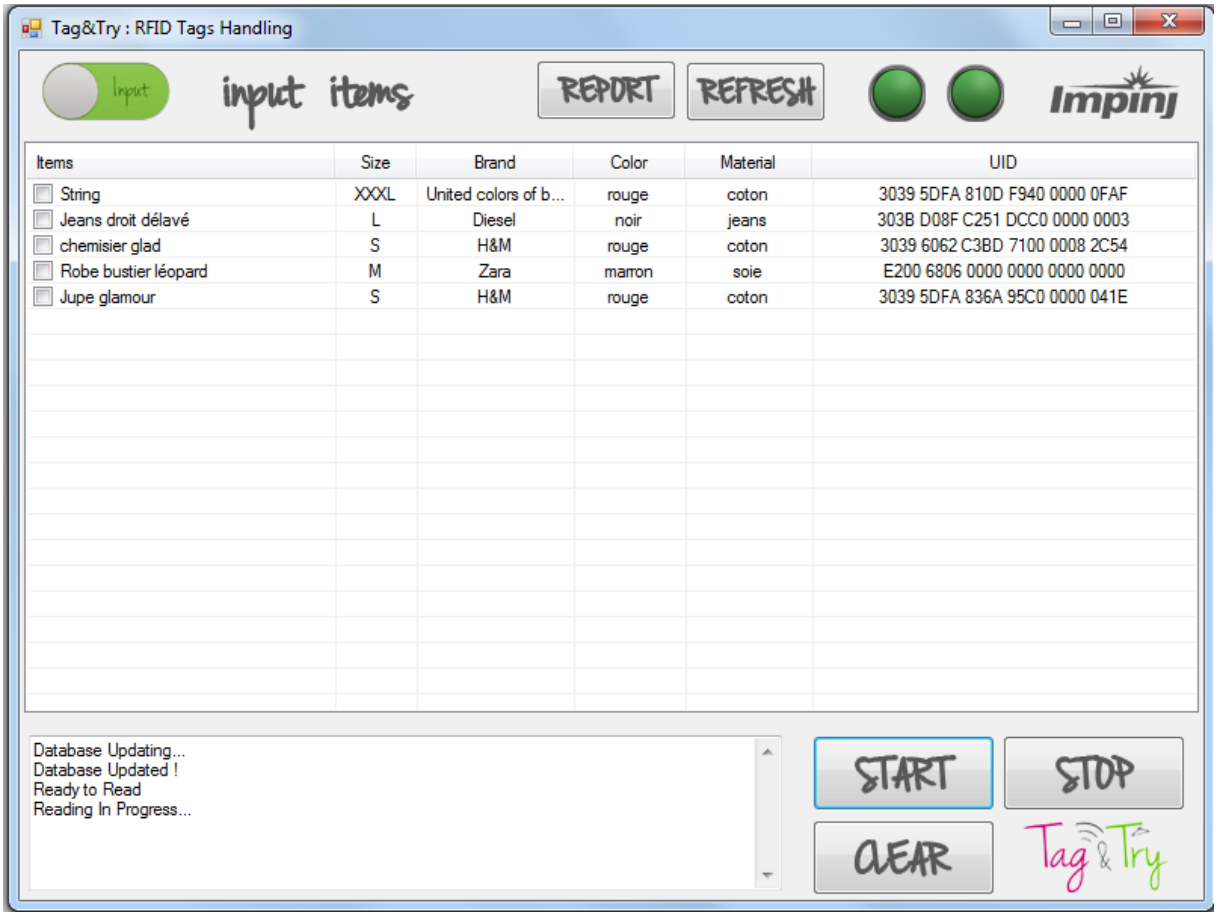


The database is automatically updated; we can read it in the console. A message of Error is displayed if there is any problem.

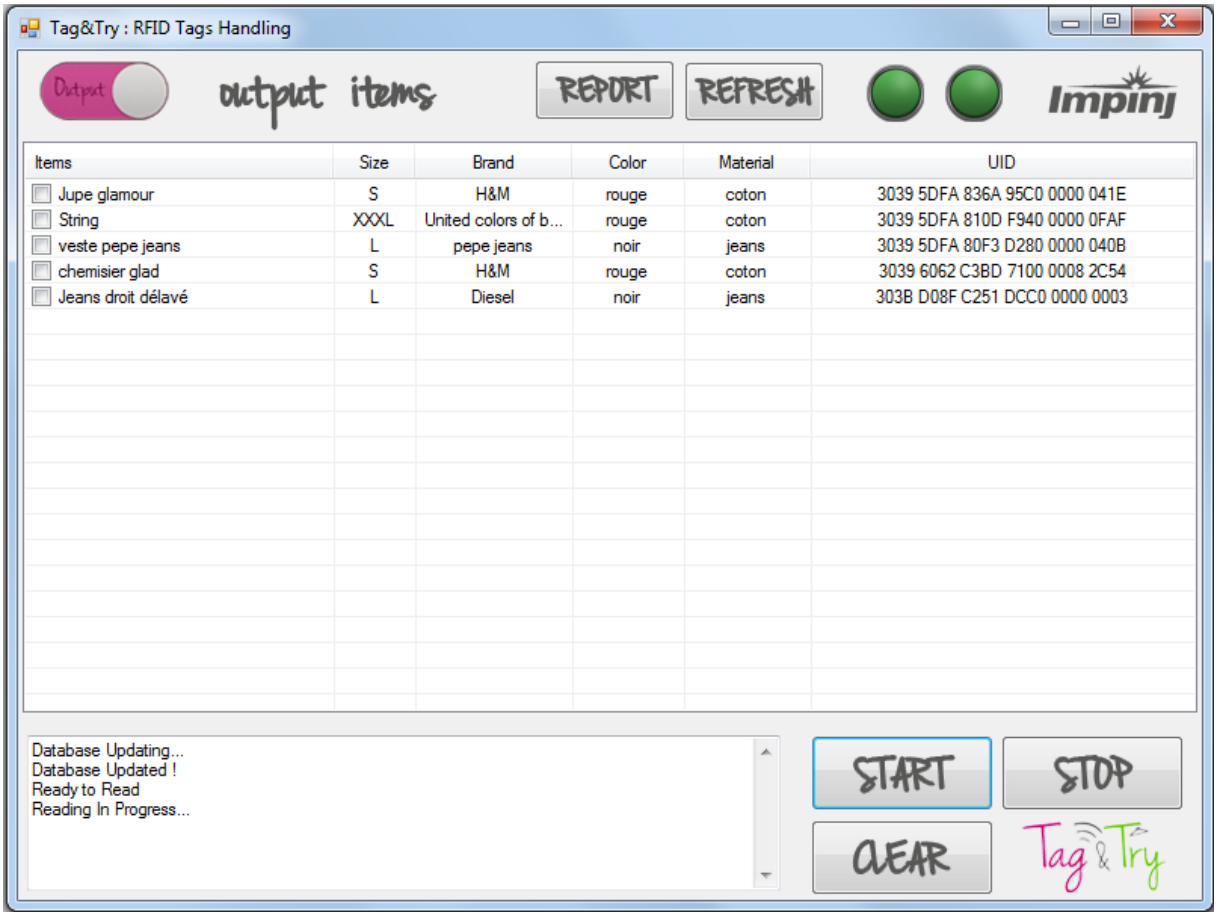
Then, you push the start button:



We can see a message in the console, showing that the reading started and the leds turned green.
When tags are reported, there are displayed:

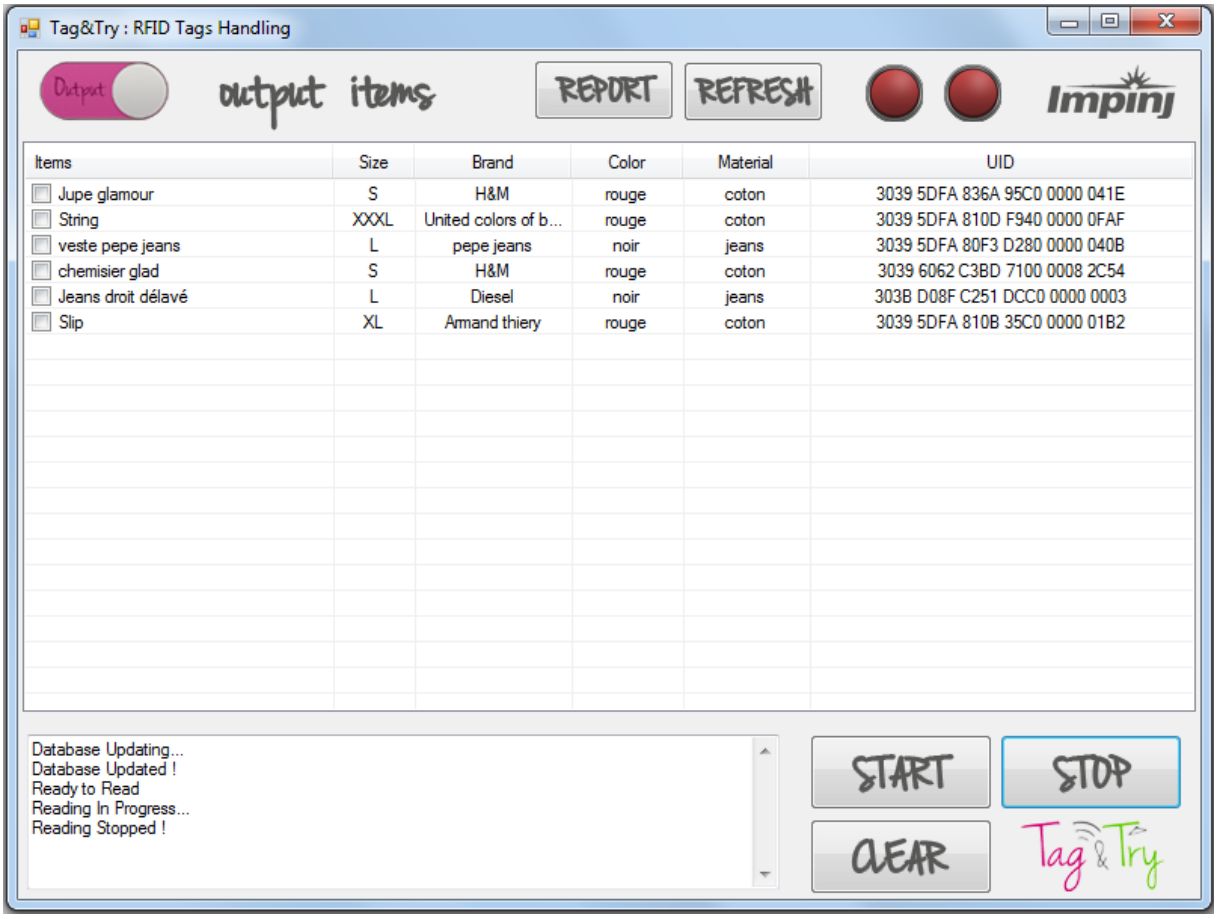


You can swith to output mode:



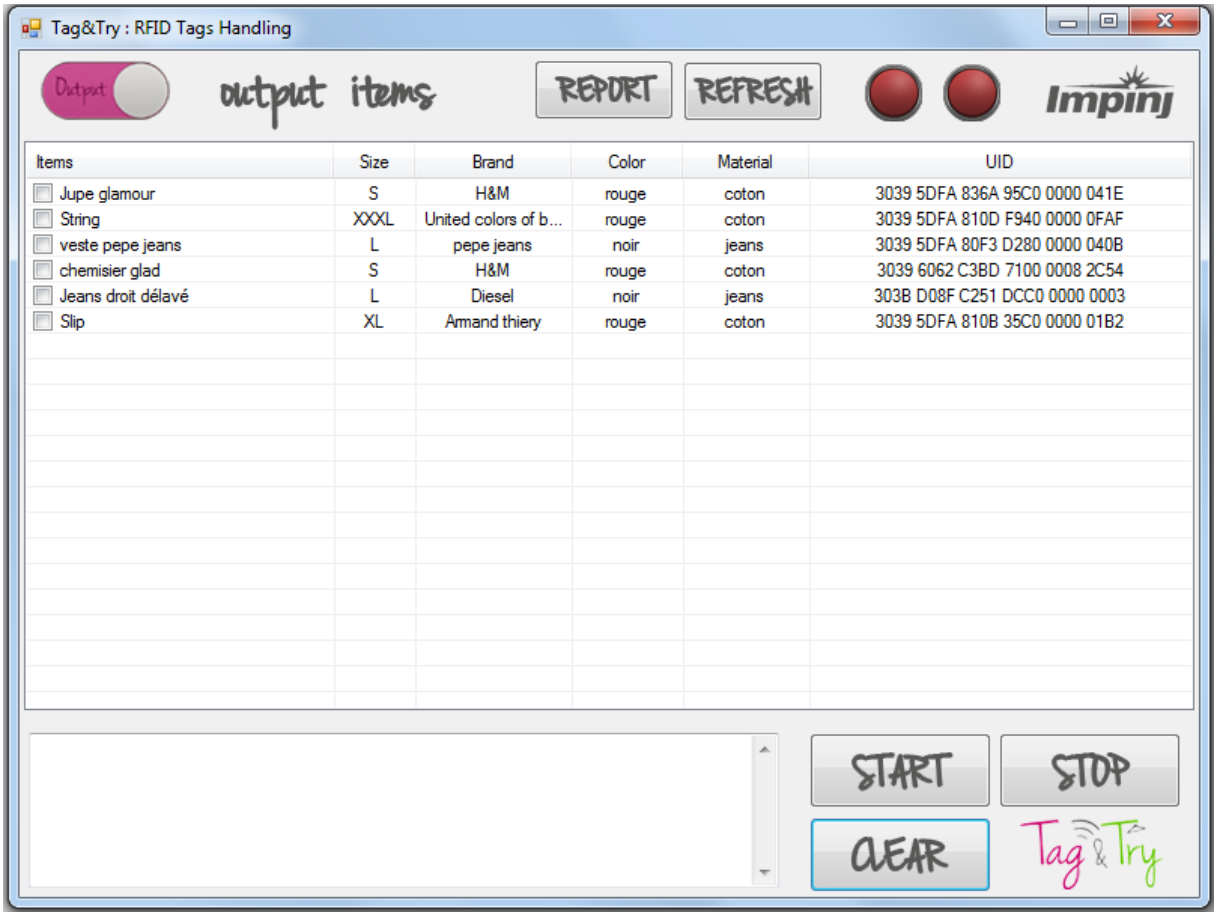
We can see that it switched to output mode, and some tags are read.

When the delivery is done, you stop the reading by pushing the stop button.



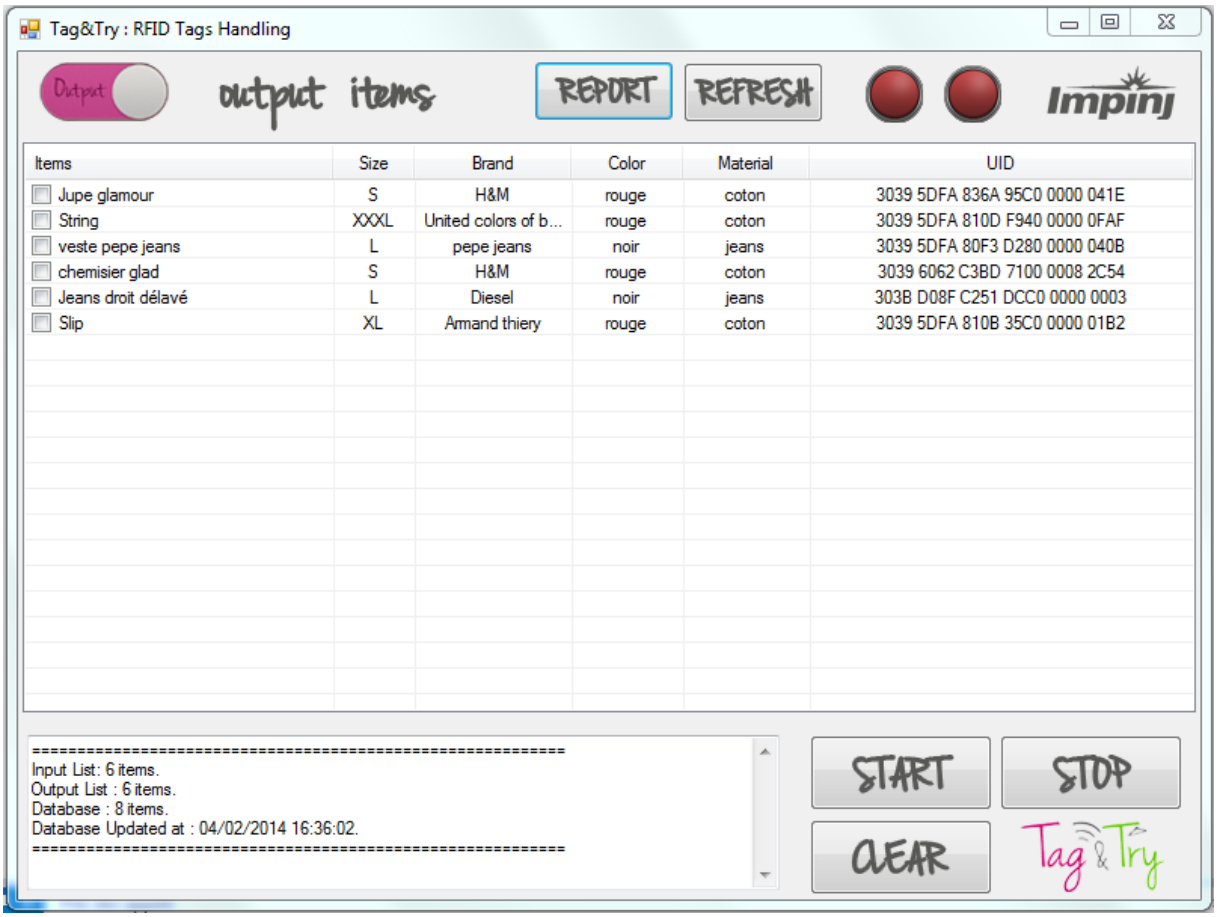
a message is displayed to show that the reading stopped, and the leds turned red.

you can push the Clear button :



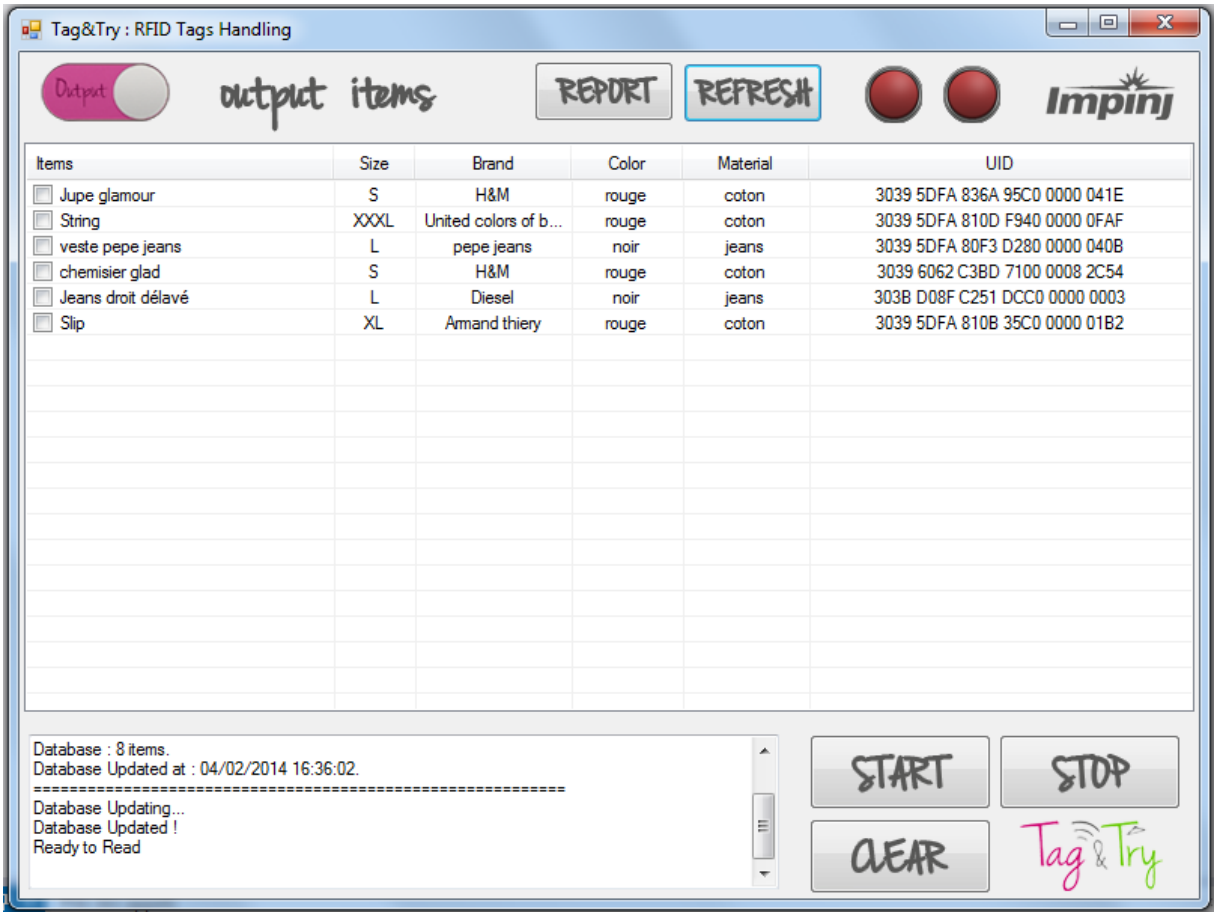
the console is immediately empty.

You can also push the report button :



It displays a list of information:
the number of items in the input list.
the number of items in the output list.
the number of items in the database.
the date and time of the last database update.

Finally if you push the refresh button:



The database is updated. And message is displayed in the console.

4.2 The technical Environments

4.2.1 The environment of the development

The project is divided into functionalities. So, we developed those functionalities in different environments.

The development environment: Bitbucket

Bitbucket is a web based hosting service for projects that use either the mercurial or Git revision control systems.

Moreover, the database is developed in Postgres. PostgreSQL, often simply "Postgres", is an open source object relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance.

In addition, the answer format of the web service is JSON. It is an open standard format that uses human readable text to transmit data objects consisting of attribute value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

Besides, we used Cordova Command Line Interface (CLI) for creating the application and deploying its to various native mobile platform.

ROR Commands are used to develop the server.

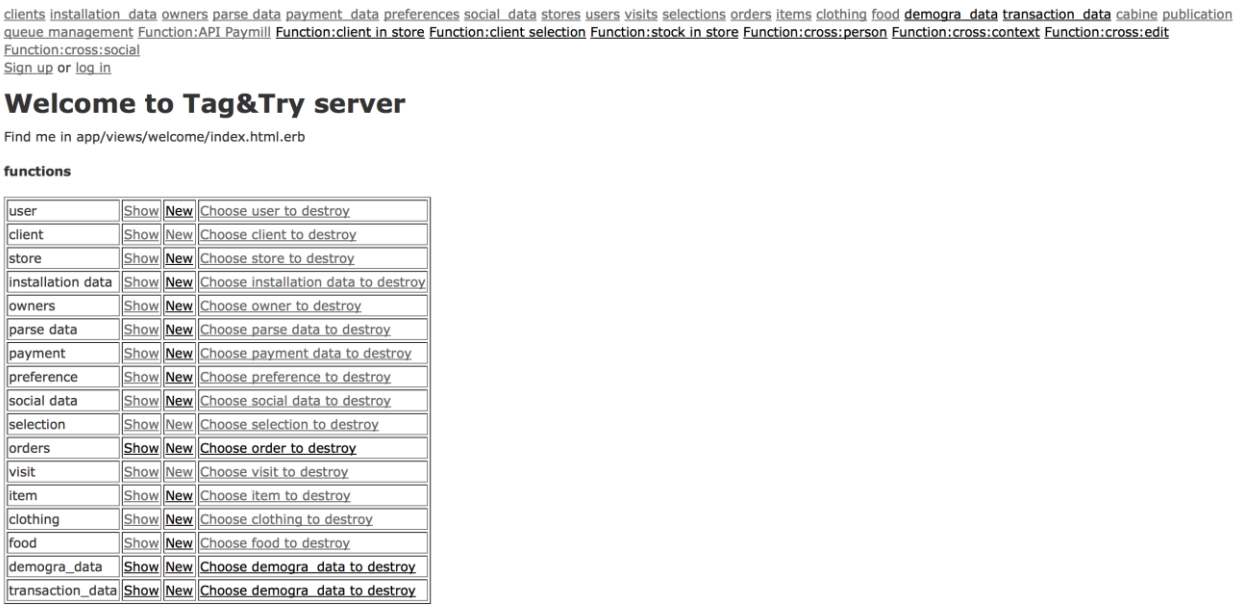
4.2.2 The environment of production

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. It can perfect support ruby on rails project. It is a platform as a service that adds simplicity to deploying, scaling and managing the infrastructure that runs applications.

Through Heroku, we can easily realize the communication between devices.

(http://tagtryserver.herokuapp.com)

For example, User has a mobile phone, they register and sent their personal data to the server, and sever save it through a Json file, then store owner application get the Json file and store the user date in its interface.



clients installation_data owners parse data payment_data preferences social_data stores users visits selections orders items clothing food demogra_data transaction_data cabine publication queue management Function:API Paymill Function:client in store Function:client selection Function:stock in store Function:cross:person Function:cross:context Function:cross:edit Function:cross:social
Sign up or log in

Welcome to Tag&Try server

Find me in app/views/welcome/index.html.erb

functions

user	Show	New	Choose user to destroy
client	Show	New	Choose client to destroy
store	Show	New	Choose store to destroy
installation data	Show	New	Choose installation data to destroy
owners	Show	New	Choose owner to destroy
parse data	Show	New	Choose parse data to destroy
payment	Show	New	Choose payment data to destroy
preference	Show	New	Choose preference to destroy
social data	Show	New	Choose social data to destroy
selection	Show	New	Choose selection to destroy
orders	Show	New	Choose order to destroy
visit	Show	New	Choose visit to destroy
item	Show	New	Choose item to destroy
clothing	Show	New	Choose clothing to destroy
food	Show	New	Choose food to destroy
demogra_data	Show	New	Choose demogra_data to destroy
transaction_data	Show	New	Choose demogra_data to destroy

We have all the functions controllers and database in server, so when other applications call some specific functions the server can always return a right Json file to show the right result.


```
[{"id":1,"User_id":5,"sizeup":"S","sizedown":"42","sizeshoes":"42","city":"Londre","country":"GB","url":"http://tagtryserver.herokuapp.com/clients/1.json"},
{"id":2,"User_id":6,"sizeup":"S","sizedown":"36","sizeshoes":"39","city":"Londre","country":"France","url":"http://tagtryserver.herokuapp.com/clients/2.json"},
{"id":3,"User_id":7,"sizeup":"XS","sizedown":"38","sizeshoes":"40","city":"Paris","country":"France","url":"http://tagtryserver.herokuapp.com/clients/3.json"}]
```

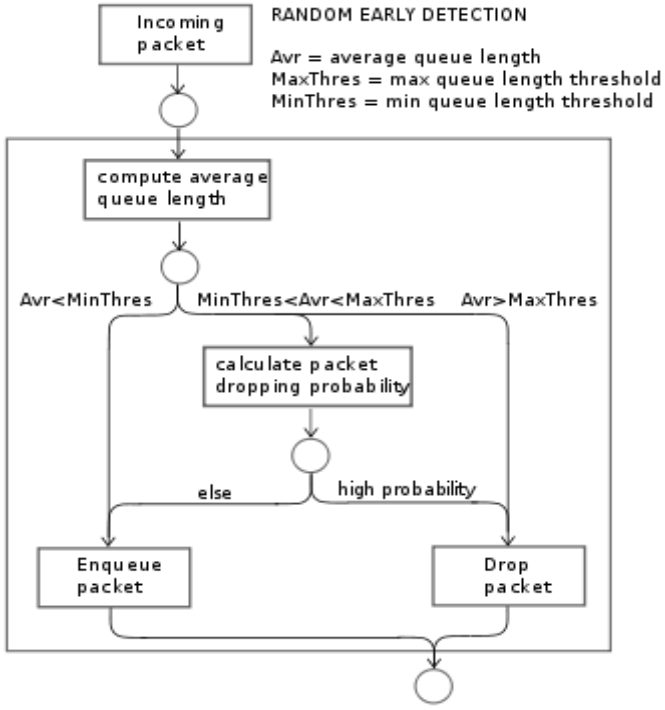
4.3 Algorithms

4.3.1 Cross selling and up selling

We developed a cross selling algorithm in ruby on rails. There are three different cross selling: editorial (the bestseller product, the new product), contextual (we recommend items with similar parameters that bought products) and personalized recommendations (color, material, theme, brand).Cross selling and up selling algorithms are systems of recommendations of product. We developed cross selling and up selling algorithms in order to recommend items to customers in store. So, we optimize the customer behavior and predict what the customer wants to buy. The recommendations may match with preferences, hobbies and social data of customers. What's more, these recommendations depend on the way of life, the opinions, the beliefs and the behavior of the customer. These algorithms will push customers to buy other products.

4.3.2 Queue Management

Besides, we implemented a queue management algorithm in ruby on rails that takes care of user management in store. We developed a queue management algorithm that displays in real time the exact position of a customer in a list and the waiting time. The algorithms may be effective in term of computing time and in term of reliability predictions. The algorithms may give good predictions. We choose RED algorithm for managing queues in store.



5 Results

5.1 System Architecture

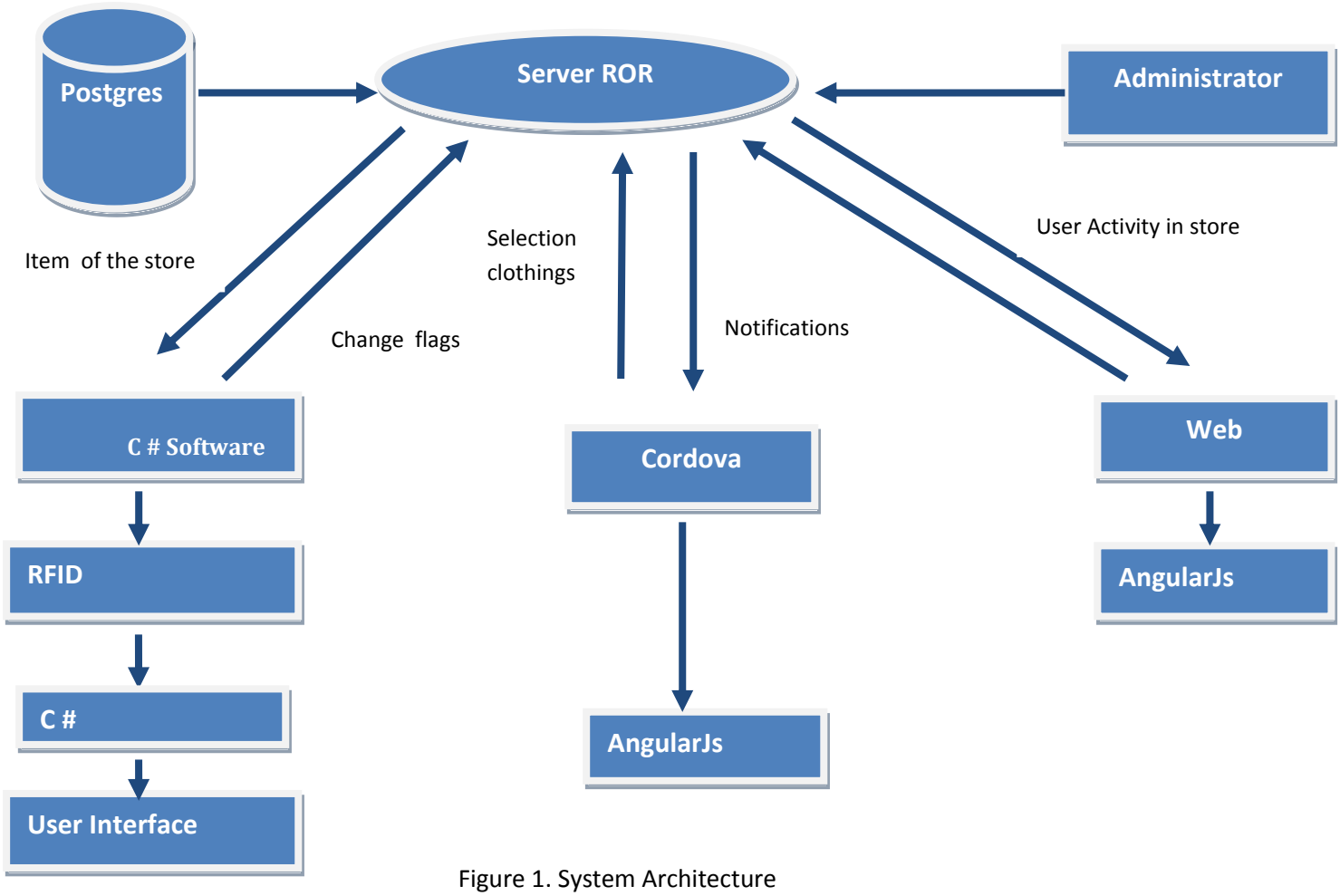


Figure 1. System Architecture

The following schema is a diagram sequence of the System:

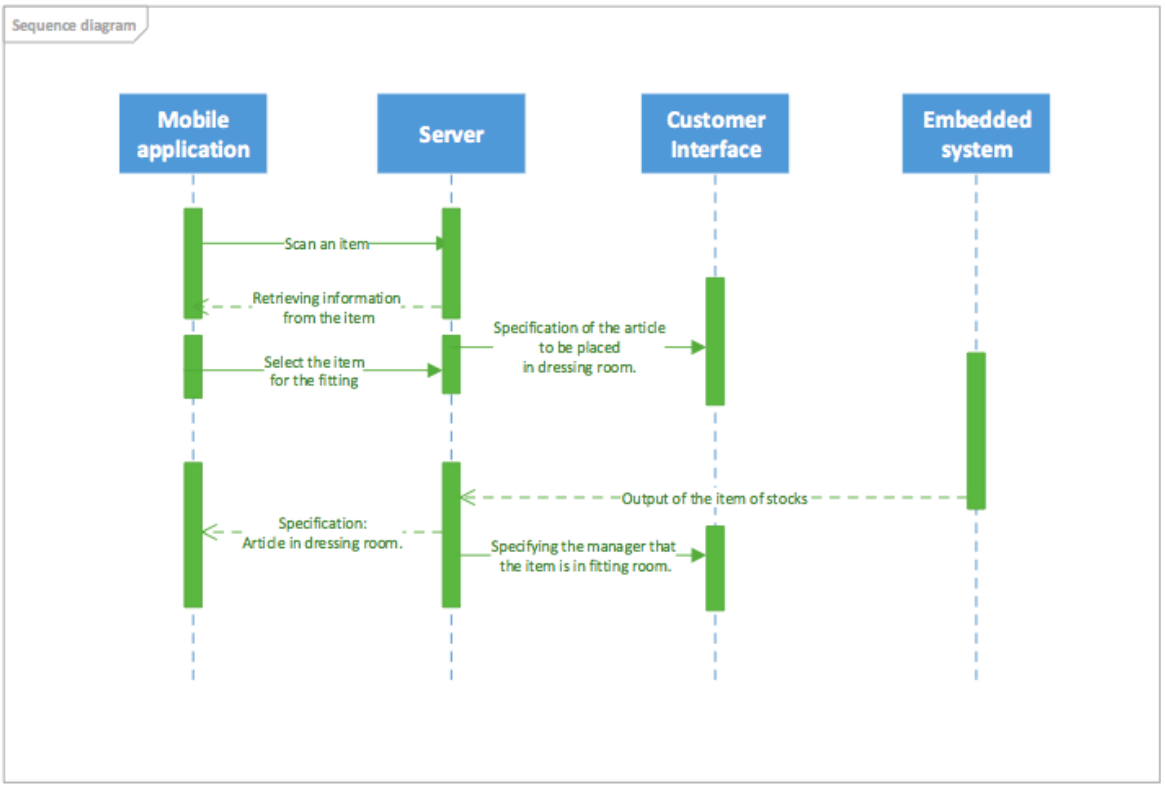


Figure 2. Diagram Sequence of the system

5.2 Server method

5.2.1 Database

We have all the data in our database, and we use postgresql, in order to better integrate with Heroku.

```
                List of relations
 Schema |               Name                | Type  | Owner
--------+-----------------------------------+-------+-------
 public | cabines                           | table | casey
 public | clients                           | table | casey
 public | clothings                         | table | casey
 public | demogra_data                      | table | casey
 public | foods                             | table | casey
 public | hahas                             | table | casey
 public | installation_data                 | table | casey
 public | items                             | table | casey
 public | orders                            | table | casey
 public | owners                            | table | casey
 public | parse_data                        | table | casey
 public | payment_data                      | table | casey
 public | paymill_on_rails_plans            | table | casey
 public | paymill_on_rails_subscriptions    | table | casey
 public | preferences                       | table | casey
 public | publications                      | table | casey
 public | queuems                           | table | casey
 public | schema_migrations                 | table | casey
 public | selections                        | table | casey
 public | social_data                       | table | casey
```

5.2.2 Function

We have a choose Controller

Inside this controller we have the following functions:
1.Get cabine state in store ex: free or busy
2.Show the number of client in store, the number of client waiting for try, the number of client trying and the number of client selected items.
3.Show cabine in a store with the client information in cabine.
4.Show all the client in store and their selections.
5.Change flag of item if the item out of stock.
6. Change flag of item if the items back to stock.

We have a cross Controller

Inside this controller we have all the function related to cross selling:
Like base on the preference of client, the server can help him or her find a item which may arise his or her interesting.

We have a mobile interface Controller

Inside this controller we have all the communication functions between server and client's application :
1.Login .
2. Get clothing by Reference.
3.caculate the time sent client to cabine.
4.Get facebook connection information.

We have a **notification Controller**

Inside this controller, we have functions for push notification, sent notification to client in order to let client go to a cabine to try clothing.

We have a **Uploader controller**

In order to save pictures that administrator upload on website.

We have a **Payment Controller**

In this controller, we focus on paymill api: we got the payment api through a gem named 'paymill', we have the following functions:

1.From Client side

-Creating a new client
-Find an existing client
-Updating an existing client only works on an instance
-Deleting a client
-For retrieving a collection of all clients

2.From payment side

-Creating a new credit card payment
-Creating a new debit card payment
-Finding an existing payment
-Deleting a paymen
-Retrieving a collection of all payments

3.From offer side

-Creating a new offer
-Updating an offer (works on an Offer instance and only the name can be changed):
-Deleting an offer
-Retrieving an offer or all offers

4.From the refund side

-Creating a new refund
-Retrieving a refund or all refund

5. From the Transactions side

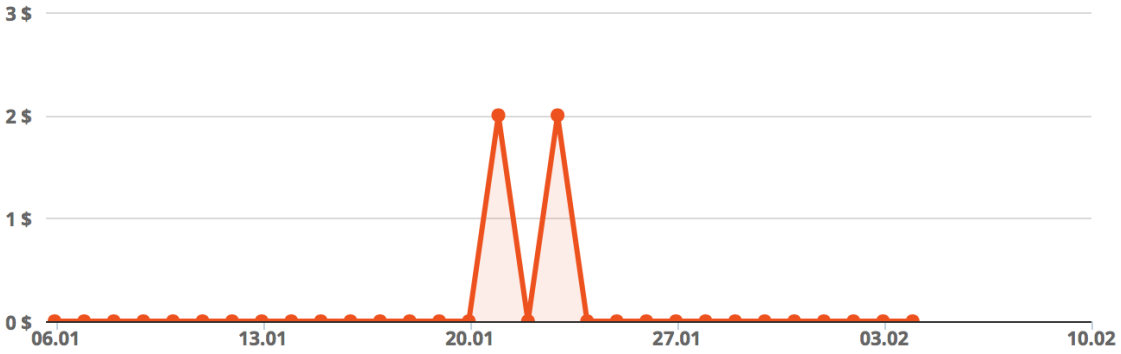
-Creating a new transaction

Demonstrate of our payment API interface on: https://app.paymill.com/fr-fr

Tableau de bord

JOUR | SEMAINE | MOIS | ANNÉE | 06/01/2014 | 05/02/2014

Volume de transaction

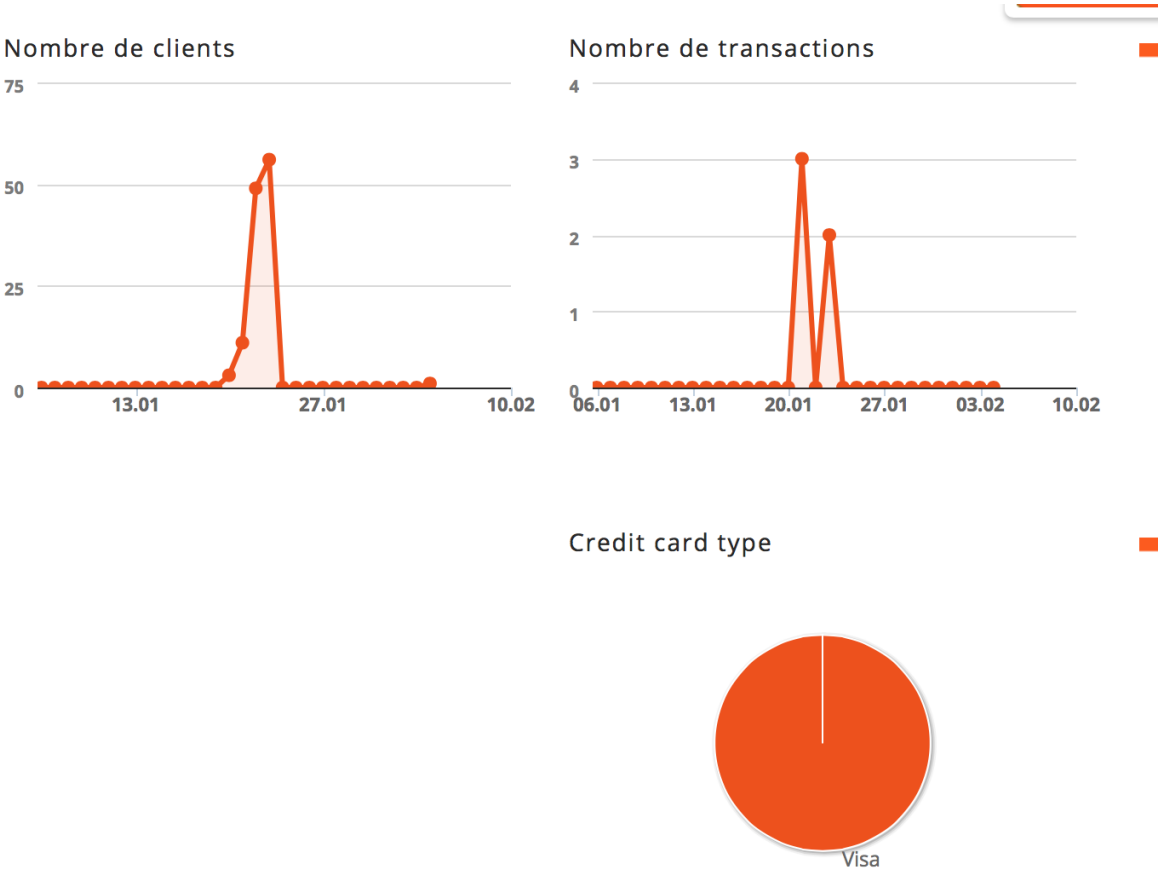


Clients	Transactions		Chiffre d'affaires		Taux de réussite
Montant	Montant	Par client	Performance	Par client	Transactions
8	5	0.63	4,00 $	0,50 $	100.00 %

Transactions

Renseigne l'ID d'une transaction

	ID	Mode de paiement	Remboursement	Montant	Statut	Dernière modification
1	tran_4ffceda05624ffd882315ced459c this is test transaction:)	VISA	-	1,00 $	●	23 janv. 2014 11:06:41
2	tran_dc70fa12306280aa8d2347743206 this is test transaction:)	VISA	-	1,00 $	●	23 janv. 2014 11:03:52
3	tran_956e8f6ad4752766a99142373b63 this is test transaction:)	VISA	1,00 $	1,00 $	●	21 janv. 2014 15:03:32
4	tran_37eb7e16e7a3d2b5f16c227b1942 this is test transaction:)	VISA	-	1,00 $	●	21 janv. 2014 14:26:13
5	tran_b965ed9c94e9848d06d6f96be2ec	VISA	-	1,00 $	●	21 janv. 2014 14:24:57

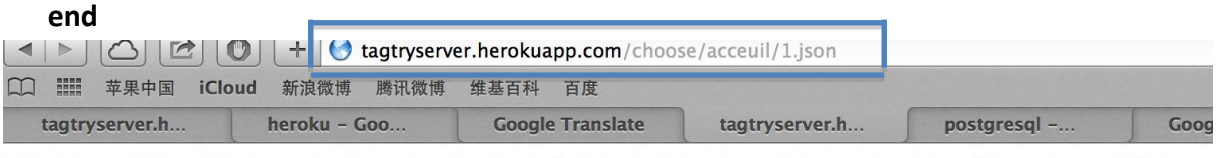


5.2.3 Render json

For each controller we need to render a json file in order to be queried.

So for each function we all use this 3 lines, in order to show data in html page.

```
respond_to do |format|
    format.json { render json:  @data}
  end
```

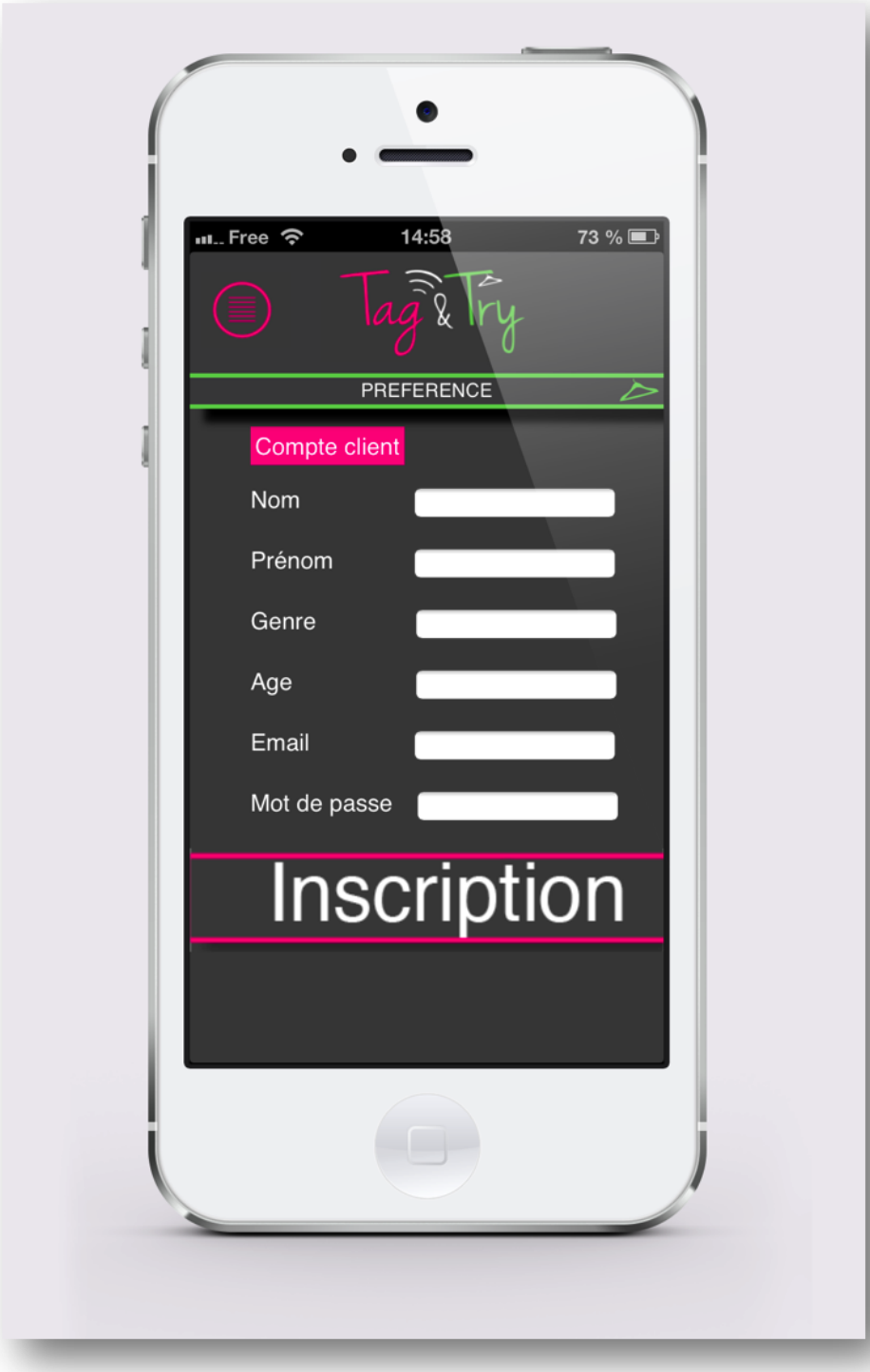


{"numberofclient":1,"numberclientinfitting":0,"numberclientwaiting":1,"numberclientselected":0}

You can see from the page that in store 1 there is 1 client who is waiting for try.

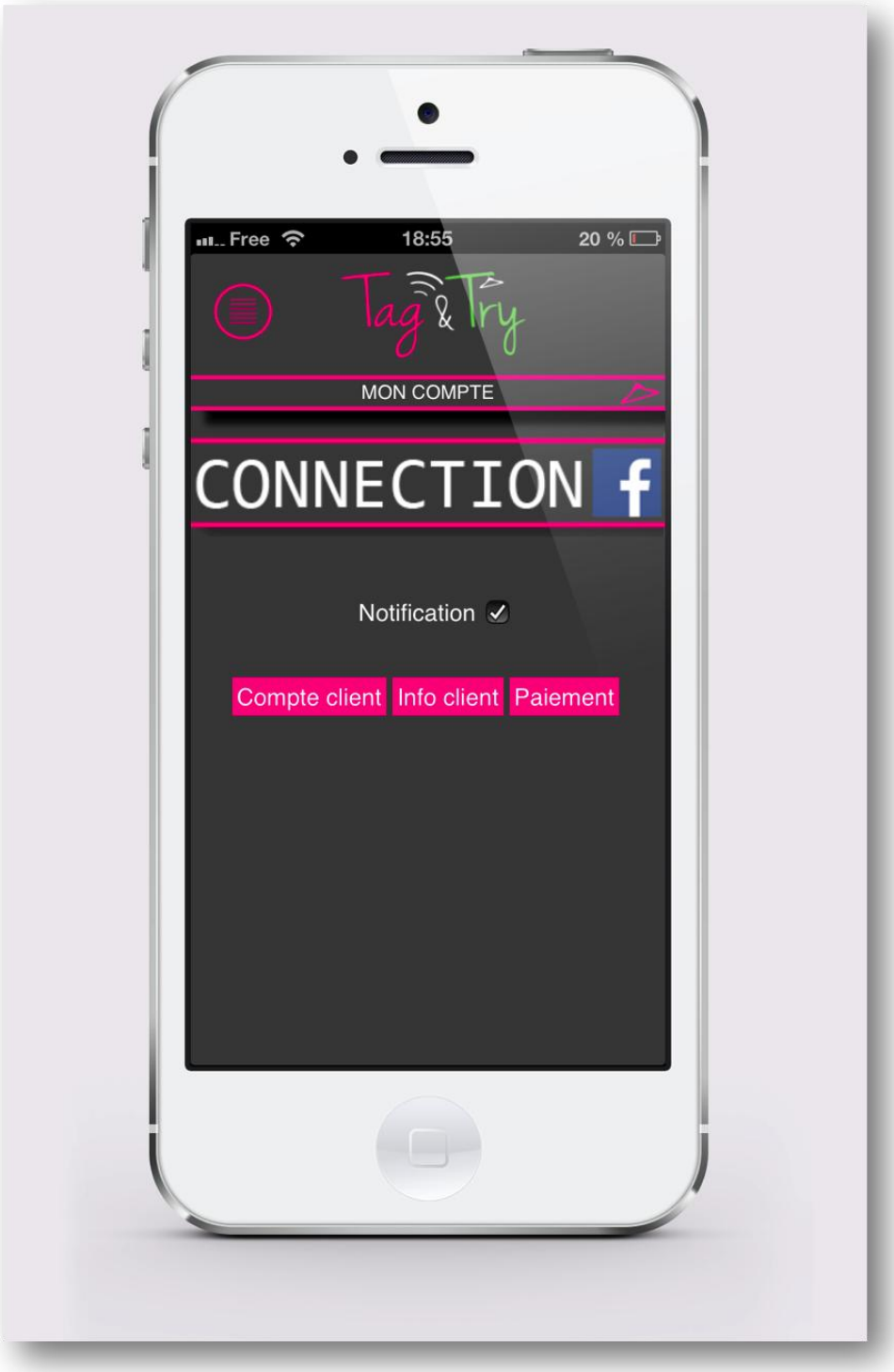
5.3 Application Tests

5.3.1 Test Functionality 1: Registration



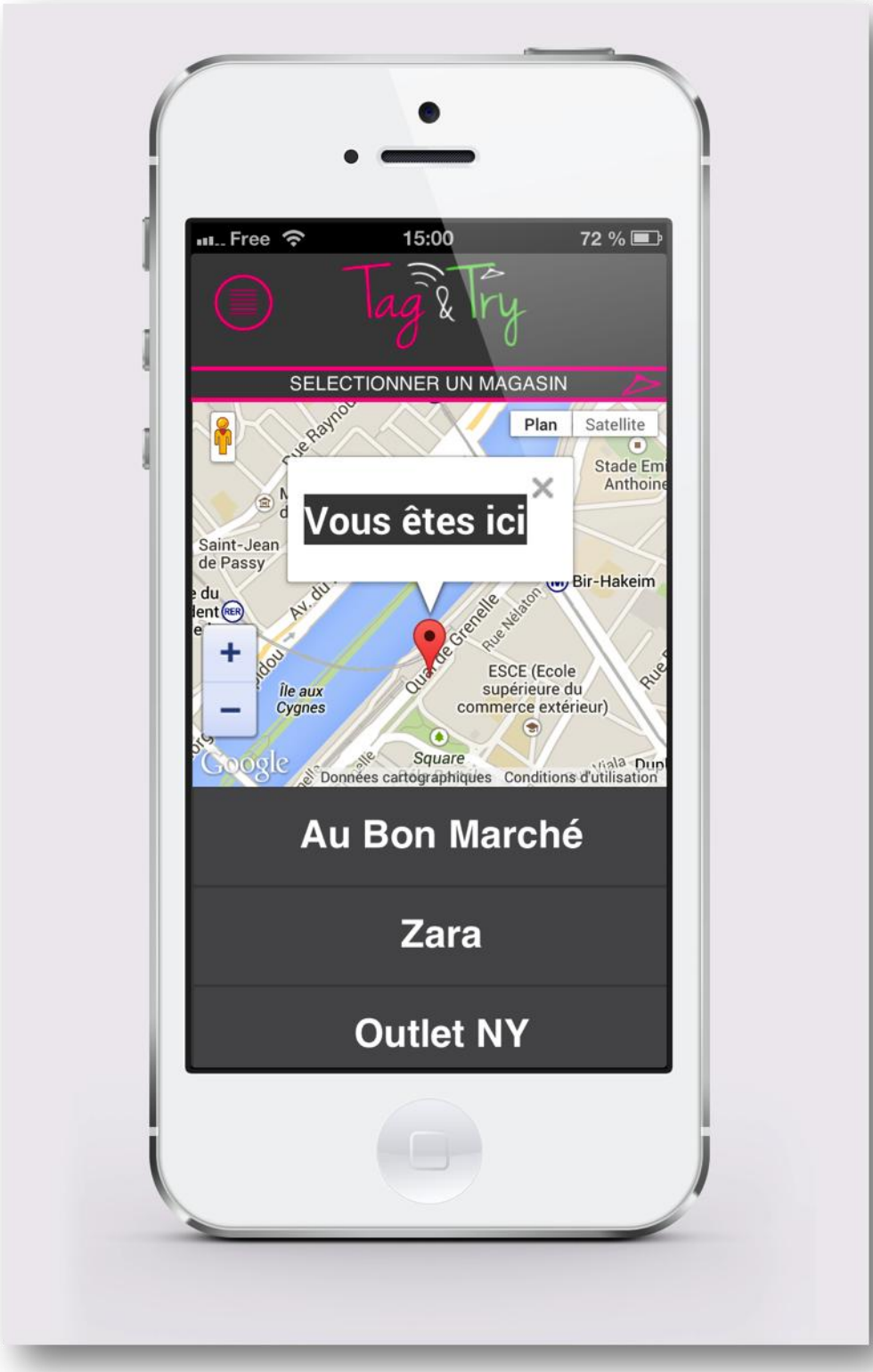
The user enter his information client, preferences (colors, brands…) and his caracteristics (size up,city, country...)

5.3.2 Test Functionality 2: Connection



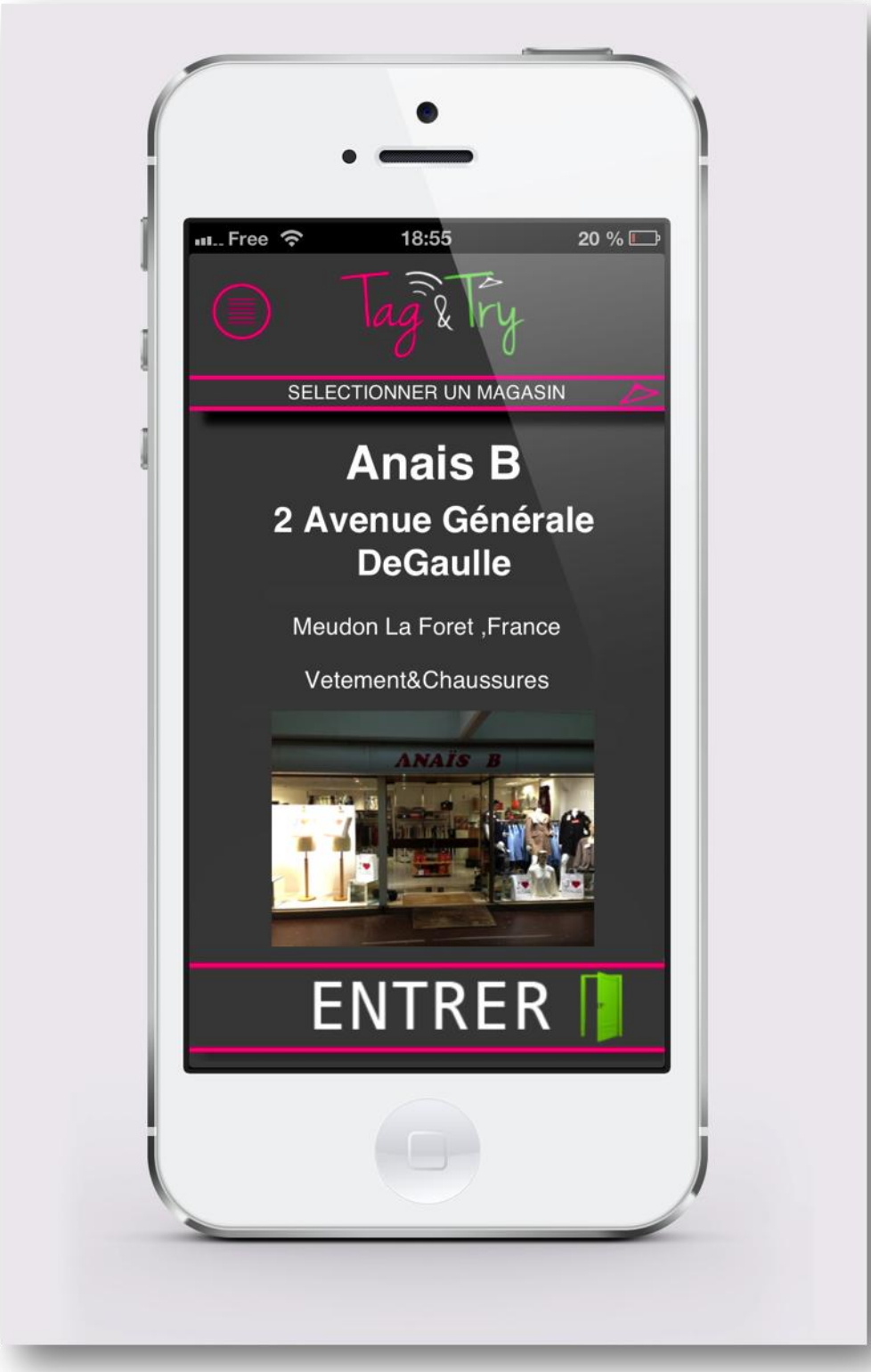
The user can be logged to the Tag&Try application with his facebook identification and can receive notifications.

5.3.3 Test Functionality 3: Store selection



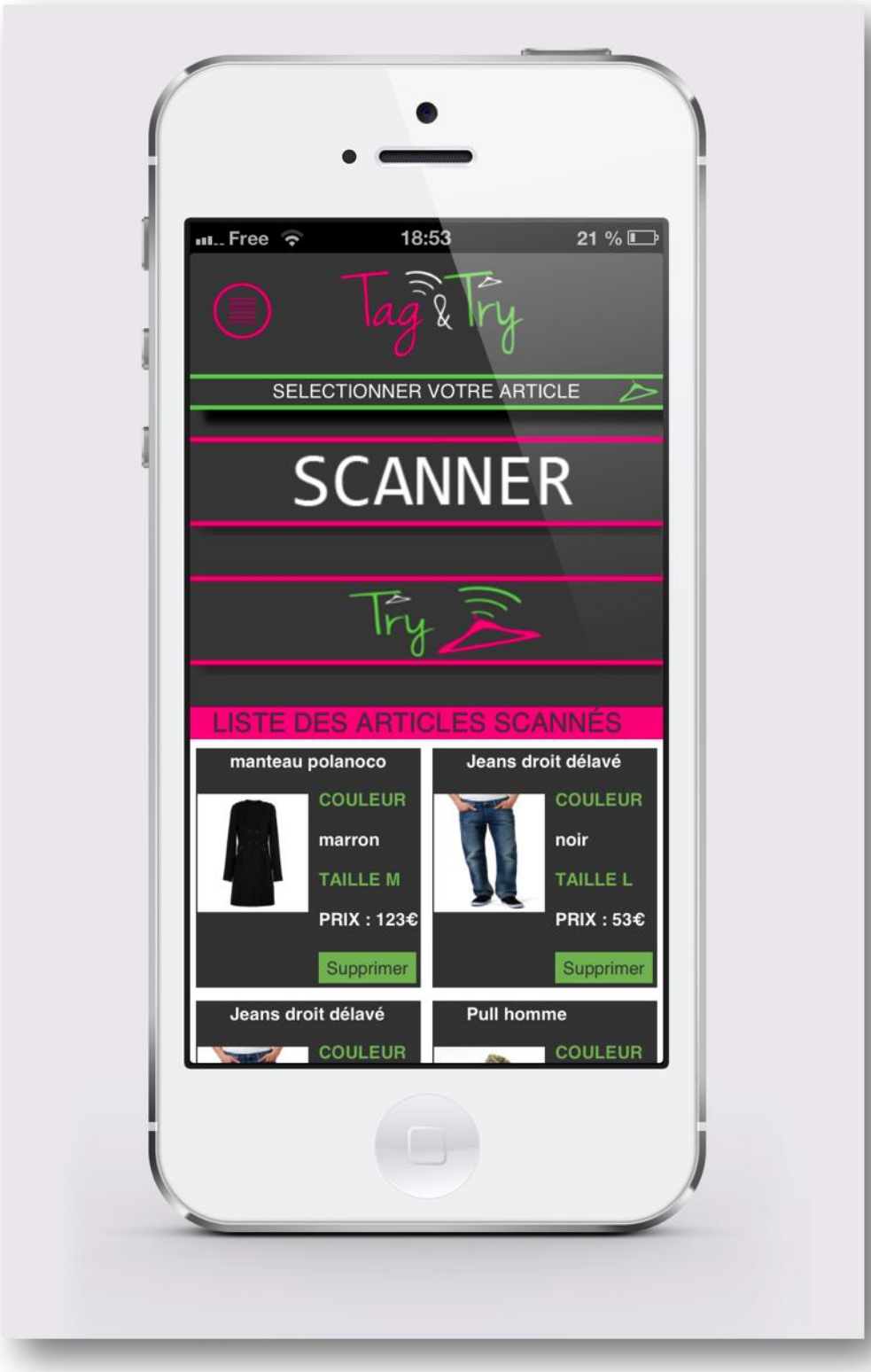
After being located, a selection of stores is suggested in the area of his localisation

5.3.4 Test Functionality 4: Store description



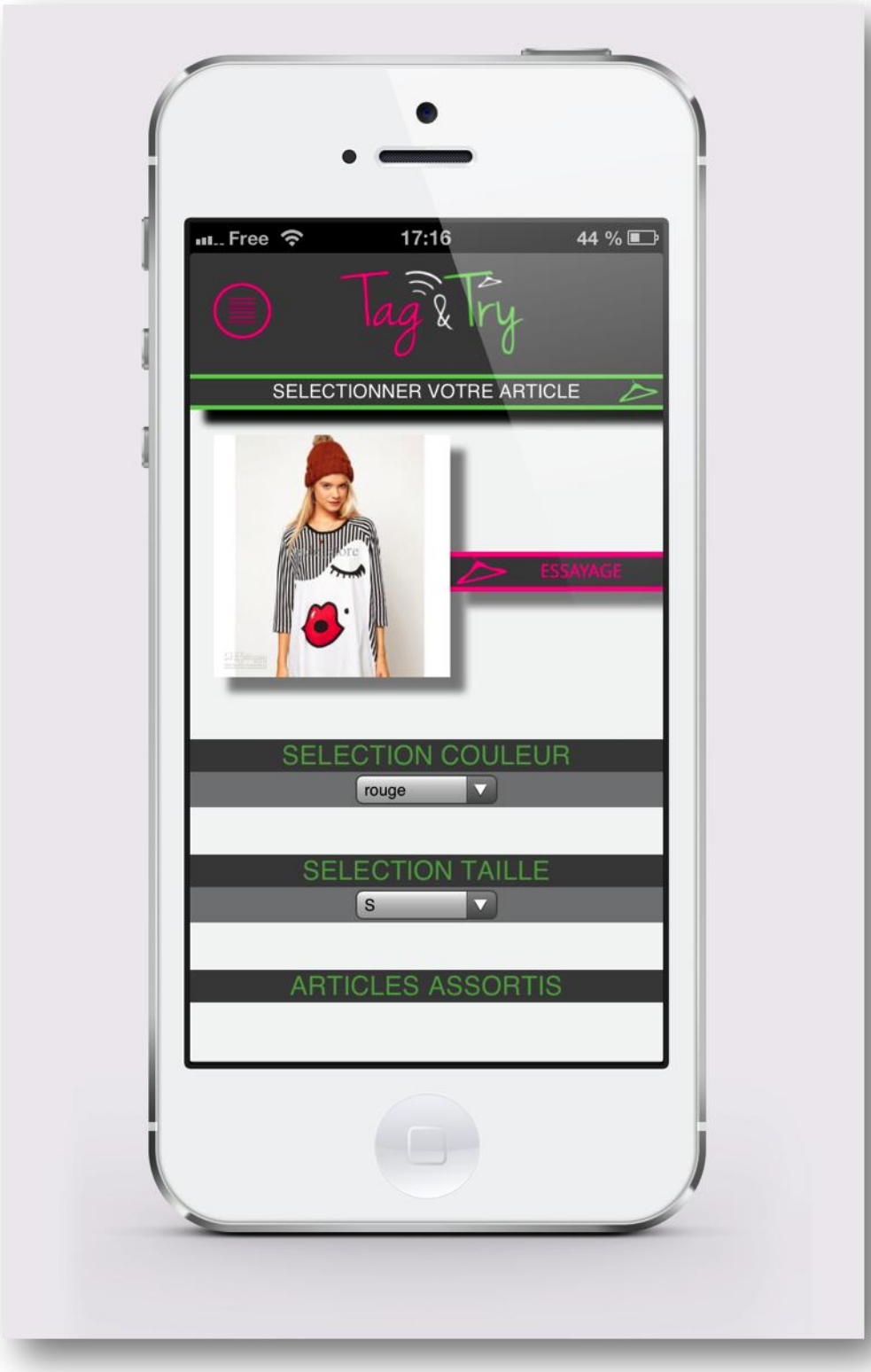
The user enters in the store to begin his clothes selection in order to try and buy them.

5.3.5 Test Functionality 5:



After having scanned all his favourite items he indicates his try intention

5.3.6 Test Functionality 6:

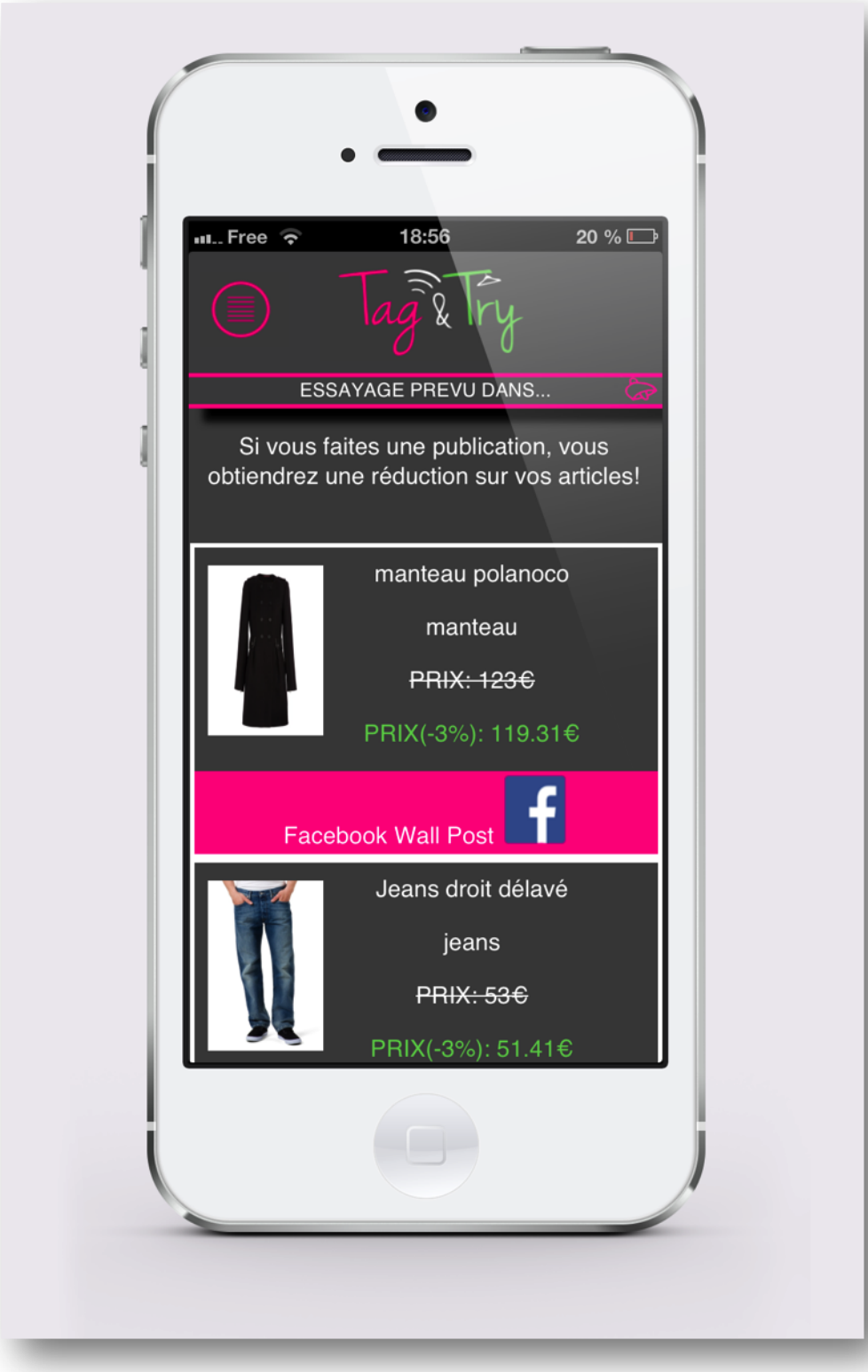


The user chooses the size and the color of the item scanned

5.3.7 Test Functionality 7:

The user has access to a scratch game during his waiting time and can benefit to a reduction on his items selection.

5.3.8 Test Functionality 8:



If the user is seduce by an item he can share it on the social network: Facebook and benefit to a supplementary reduction.

5.3.9 Test Functionality 9:

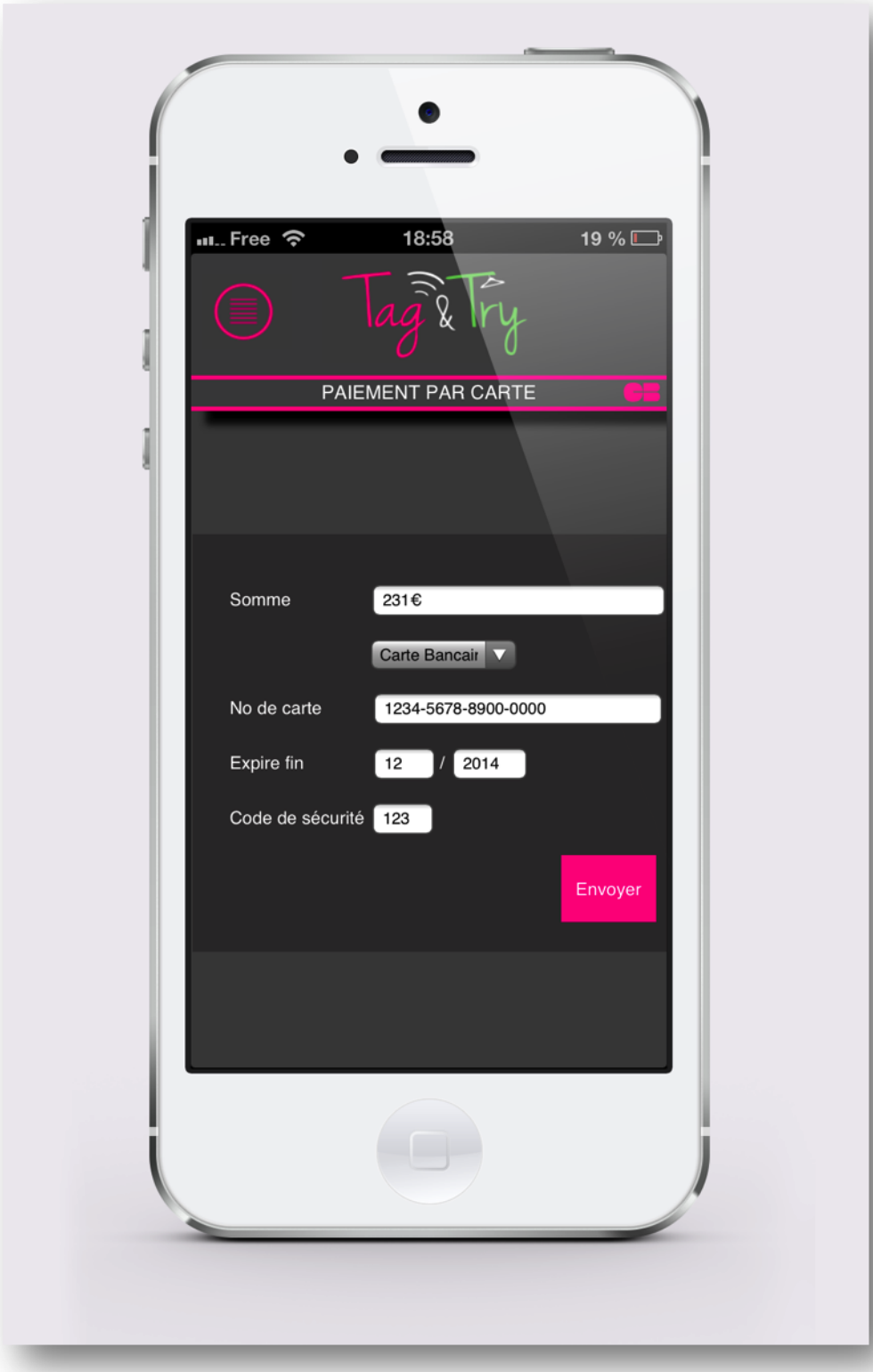
The item is share on Facebook and all the user's friends can exchange their opinions.

5.3.10 Test Functionality 10:

The list of items selected is displayed before the payment

5.3.11 Test Functionality 11:

5.3.12 Test Functionality 12:



5.3.13 Test Functionality 13:

5.4 Manager interface

The manager interface allows knowing all what's going on in the store in real time. Information used is retrieved via all information returned by the user's mobile application store.

this application is made in html and CSS and we use many features of angularJS to retrieve the information and work on the server requests. In fact we had .Json files on the server, these json files were create as one goes along that information were send to the server.

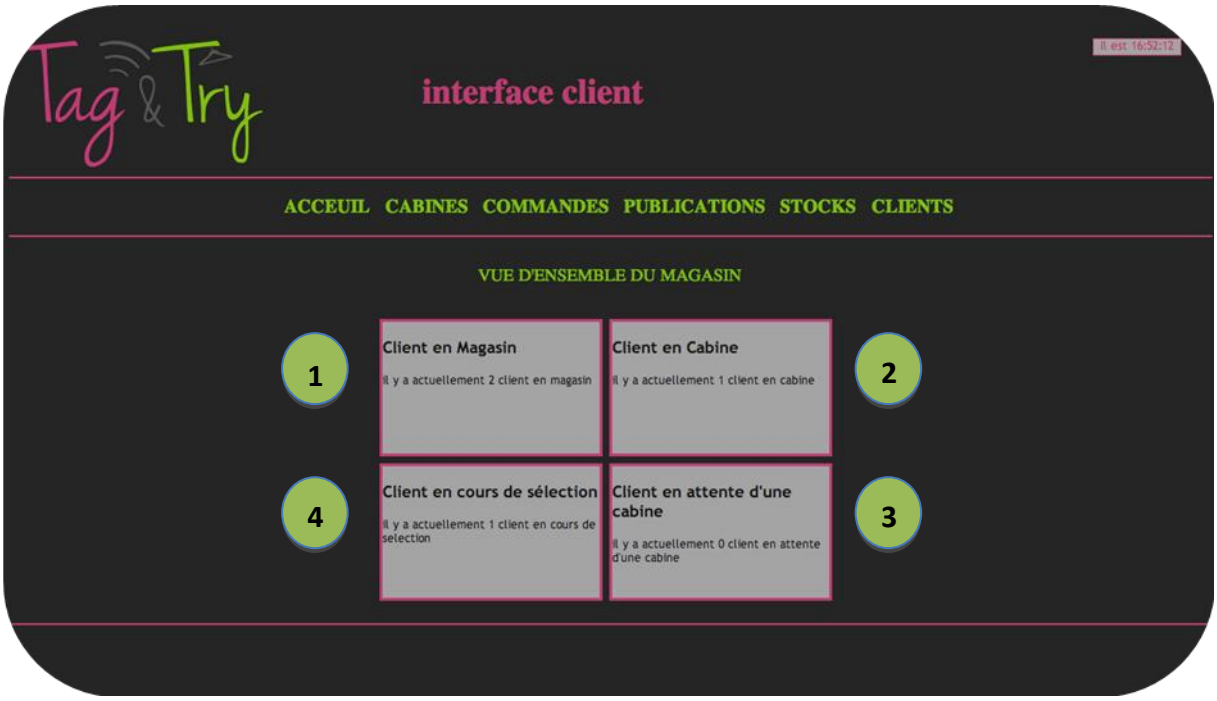
To get back information from the serveur, we have to make some request in the manager interface thanks to a controller and a services platform. We stock some information in this controller and we call this one in the html code.

The interface is divided into several steps

5.4.1 Homepage

In the homepage we have access to those information :

1. How many customers are in the store.
2. How many customers are in fitting room.
3. how many customers are selected articles.
4. How many customers are waiting for a fitting room.

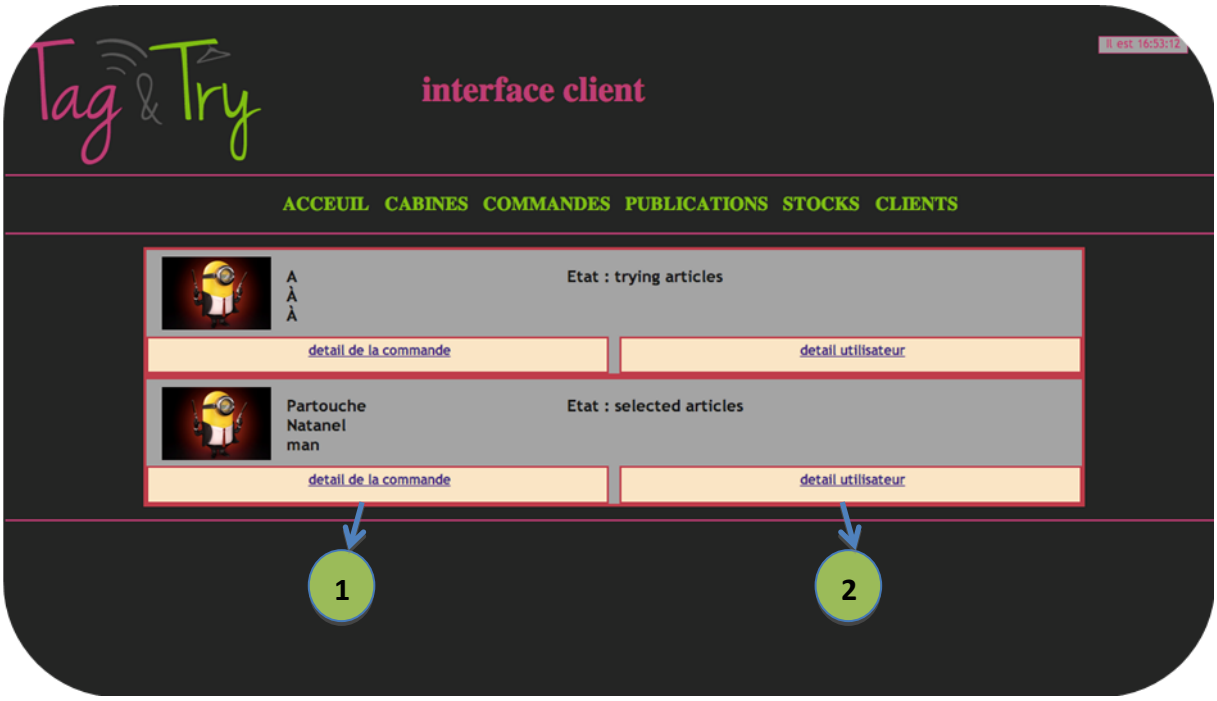


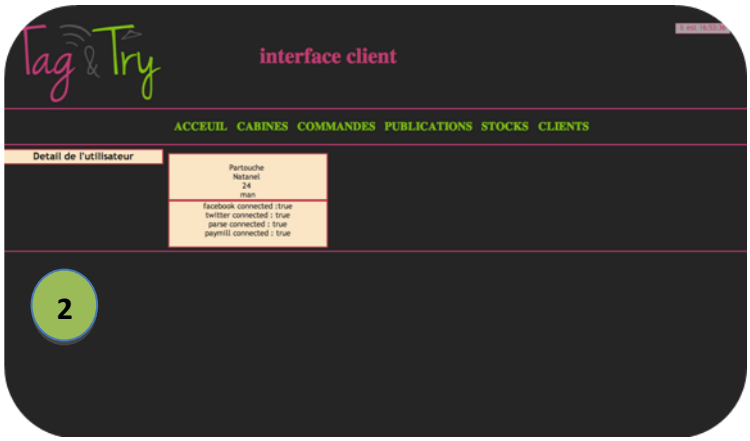
5.4.2 Customers Page

A "clients" page where all customers in store are referenced with several features :

- Photo.
- Name.
- Firstname.
- Gender.
- State (if it is under selection , trying articles, waiting for a fitting in fitting room , or doing nothing).

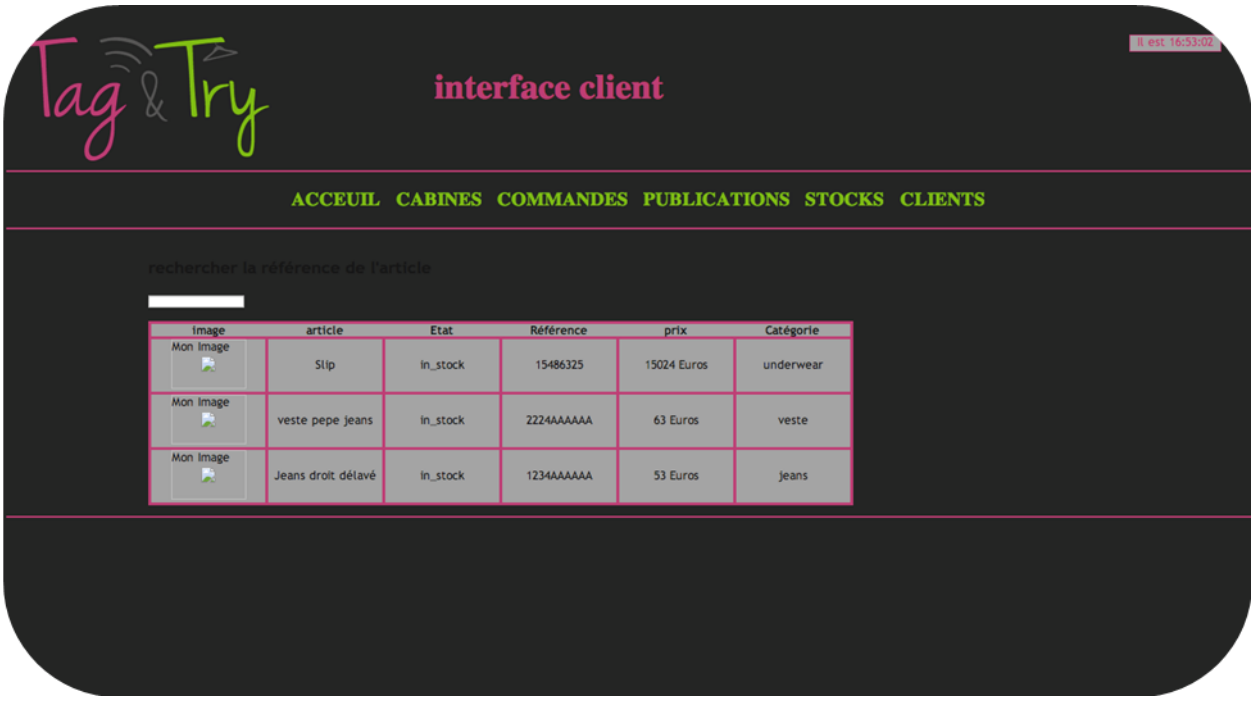
We create two more buttonswitch can alow with a simple clik to acess to the details of the user and the user order.





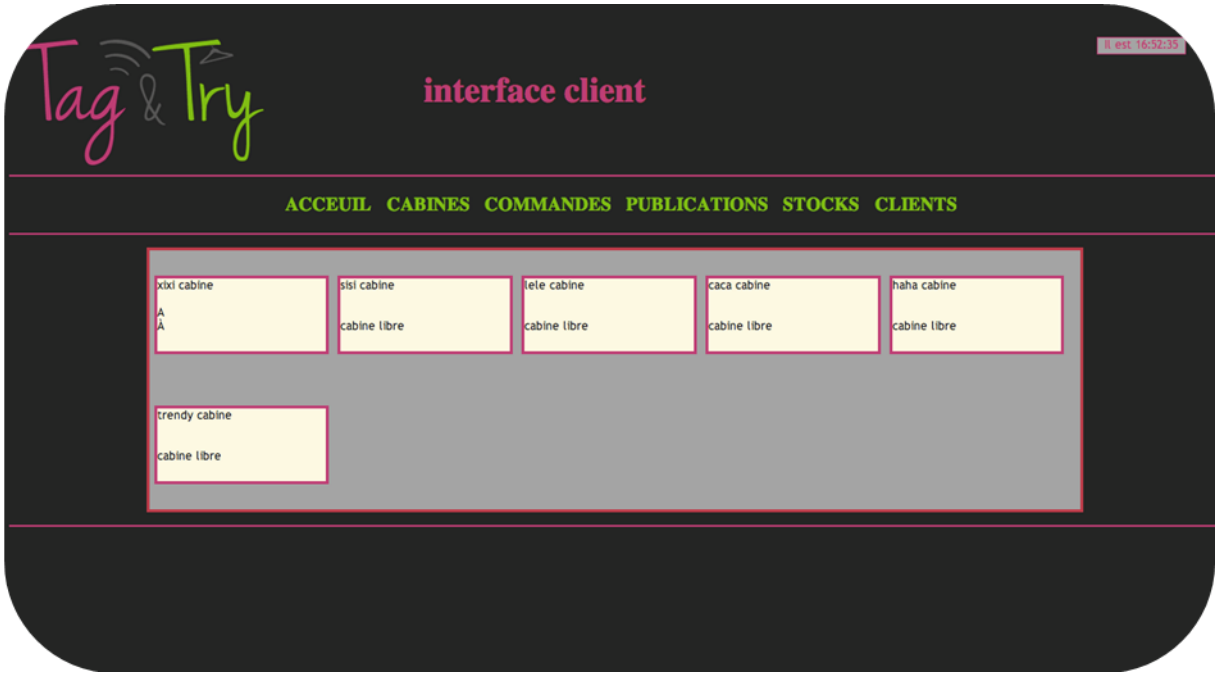
5.4.3 Stocks Page

Stock page allows to know all the stock available in the store.
it is also possible to search each item by its reference, its title, or its price. This feature is achievable through AngularJS.



5.4.4 Fitting-room Page

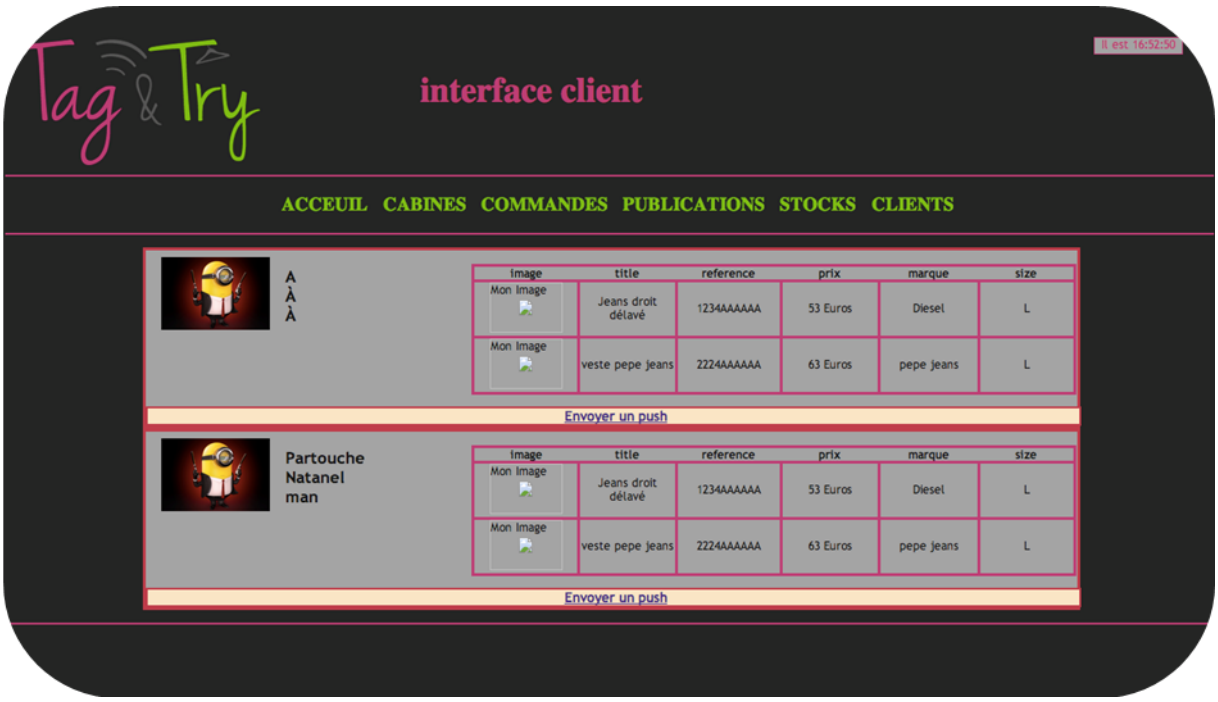
The page fittings-rooms lets us know which user is in the cabin.
If a fitting-room is not available it displays the name of the user who is inside. Otherwise the fitting-room is displayed as free.



5.4.5 Order page

The order page allows referencing all orders. When the client requests the fitting of these items, his status changes from "selected" in “waiting". We can then prepare the order of the customer, and when his command is ready, send him a push notification on his Mobil

phone to tell him that and he can go to try all these items.



6 Difficulties encountered

First, we have had difficulties during the integration of PhoneGap /Cordova. But, José helped us to solve those problems because he gave us advices for the integration of PhoneGap /Cordova.

At the beginning, we don't which framework UI we have to choose: Jquery or Jquery Mobile.

Everybody in the group doesn't have the same knowledge in informatics, so some of us had to learn every-thing to do their part of the project. For Example, to developed the manager interface, Pierre-Alexandre had to learn the html and css language, and after that he had to understand how we can use angularjs in his part. But this learning was possible thanks to the support of the group.

In addition, we encoutered problems for:

- channeling features (posgres Development)
- the connectivity unit
- import plugins
- cross domain
- payment
- push
- facebook
- the database

Despite those difficulties, we were able to face to them by working very hard and by asking to expert's advices. Thanks to our spirit team, we managed to overcome our problems. In fact, Henry Ford said:
« coming together is a beginning; keeping together is progress; working together is success ».

7 Conclusion : ways of improvements

By way of conclusion, we designed a mobile application that meets to the specifications. So, the specifications are reached. Indeed, this android application allows to share, play, care and value the wait time promotions. We managed to develop a

We can improve our solution in particular for the mobile integration (RFID NFC, bluetooth).

8 Glossary

ROR: Ruby On Rails is an open source web application framework which runs on the Ruby programming language. It is a full-stack framework: it allows creating pages and applications that gather information from the web server, talk to or query the database, and render templates out of the box. As a result, Rails features a routing system that is independent of the web server.

MVC: Model View Controller is a software pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

PHP: Personal Home Page is a server-side scripting language designed for web development but also used as a general-purpose programming language.

HTML: HyperText Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.

CSS: Cascading Style Sheets is a style sheet language used for describing the look and formatting of a document written in a markup language.

NFC: Near Field Communication is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into proximity, usually no more than a few inches.

JSON: JavaScript Object Notation is an open standard format that uses human-readable text to transmit data objects consisting of attribute value pairs.

9 Bibliography

1. D. Lin and R. Morris. *Dynamics of Random Early Detection.* October 1997.

2. Ford Harding. Cross-Selling Success: *A Rainmaker's Guide to Professional Account Development.* October 2002.

3. Sally Floyd, Ramakrishna Gummadi, Scott Shenker. *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management*. August 2001.

4. Ningning Hu, Liu Ren, Jichuan Chang. *Evaluation of Queue Management Algorithms*. January 2001.

5. Floyd, Jacobson. *Random Early Detection gateways for Congestion Avoidance.* August 1993.

6. W. Feng, D. Kandlur, D. Saha, K. Shin. *Blue: A New Class of Active Queue Management Algorithms*. April 1999.

7. Ruby on rails tutorial http://guides.rubyonrails.org

8. Paymill documentation https://www.paymill.com/fr-fr/documentation/