

Projet de Concept pour la création et la
réalisation de magasin en accès libre-
service avec promotion en temps réels



4PJT Supinfo

SupMagasin

M.Sc.1 promo 2019-2020 Grenoble

4PJT



Documentation Technique Projet Sup Magasin

I) Stack Technologique

L'architecture du projet sera de type back/font end.

Constitué d'une api en C# ASP .NETCORE dans sa version 3.1. Ce choix est dû principalement aux connaissances avancées dans ce langage de nos développeurs. Elle aura le rôle central dans la solution, car elle servira d'interface avec la base de donnée(BDD), et les différents éléments nécessaires à son utilisation.

Le front, web et mobile se fera en Ionic version , Framework javascript frontend, couplé à du typescript par le fait de notre développeur frontend plus a l'aise avec cet outil.

L'API et le front communique par le biais d'appel front et de renvois de données sous format Json. La sécurité est assuré par des JWT(javascript web token), dérivé de l'auth2.0. Les tokens sont valables 24H et nécessaire pour chaque requête sur l'api sauf (route User).

L'Api est une interface entre le base de données, mis en place en Mongo DB dans sa version 5.2?. Pour sa souplesse et son économie de ressource par le stockage de données dans des fichiers JSON.

L'Architecture a été pensé pour être le plus souple possible. Nous avons décidé d'utiliser Windows Sever 2016 pour l'hébergement et la création du réseau au sein de l'application.

Les bornes Bluetooth sont pour ce POC, des razberry pi 3B, montée par nos soins, pour vérifier et identifié les usagers dans le magasin.



II) La Base de Données (annexe 1)

La base de données a été pensée pour être la plus compact possible en rassemblant les éléments de Type 1/N dans un seul schéma.

Par cette méthode, nous avons pu dégager 4 schémas, le Customer qui représente la pierre angulaire, comprenant les différents éléments liés aux clients, de ses moyens d'identification à ses données personnelles.

De plus nous l'avons liée au schéma vente afin que celui-ci soit liée aux produits qu'il achète et que nous utiliserons dans la recherche de promotion en temps réels dans le schéma produit, tous en conservant une historique des ventes.

Ce dernier schéma cité comprend toutes les informations de produits ainsi que ces stocks et son fournisseurs attiré. La réflexion c'est porté, qu'un seul fournisseur ne pouvait produire qu'un unique produit et que nous ne passions que par lui pour les commandes.

Les stocks sont pris en comptes avec une alerte signalée en cas d'atteinte d'un stock d'alerte et d'une menace de rupture de stock.

De plus nous assurons la traçabilité des produits par l'enregistrements des lots, (nous achetons à des grossistes) avec dates d'achats et dates de péremptions pour les produits périssables (Alimentaire)

La table magasin quant à elle, dans ce POC rien n'était indiqué pour la présence de plusieurs magasins, donc nous avons préféré prendre de l'avance et anticiper des ouvertures futures. Elle comprend les données liées aux magasins ainsi que ses employées que nous affectons à chaque magasin. Ces dernières données comprennent, les informations liées à chaque employé ainsi que ces logins pour ce connecter à la partie caché du magasin et comprend différentes fonctions de gestions (contrôle en temps réels des stocks, alertes, gestions des rayons et commandes, monitorings, etc.)



III) Les Routes

L'ensemble des routes se trouvent sur la documentation swagger à cette adresse, ainsi que ses entrées valides :

https://app.swaggerhub.com/apis/Zamiquie/apisup_magasin/v1

Voici un aperçu :

L'URL de l'api:

<http://apisupmagasin.ddns.net>

Customer ^	
GET	/Customer/All
GET	/Customer/{id}
GET	/Customer/{id}/BanqDatas
GET	/Customer/{id}/Phones
POST	/Customer/addCustomer
POST	/Customer/addMultiCustomer
PUT	/Customer/updateCustomer
DELETE	/Customer/Delete
Default ^	
GET	/
OPTIONS	/
Error ^	
GET	/Error
Product ^	
GET	/Product/All
GET	/Product/{id}
GET	/Product/name/{designation}
GET	/Product/{id}/commentary
GET	/Product/{id}/supplier
GET	/Product/{id}/deliverv



IV) L'authentification

L'authentification est enregistrée dans le Schéma Customer.

a) L'enregistrement (<http://apisupmagasin.ddns.net/user/create>) : POST

```
POST http://apisupmagasin.ddns.net/user/create

{
  "Id": null,
  "Sexe": 1,
  "Email": "corentin.cannie@wanadoo.fr",
  "Password": "azerty38",
  "Name": "Cannie",
  "FirstName": "Corentin",
  "BirthdayDay": "0001-01-01T00:00:00",
  "Adress": "25 rue de la coquellettes",
  "Postal_Code": "38580",
  "City": "St Vincent en Vigne",
  "BanqData": null,
  "Phones": null,
  "Photo": null,
  "Last_Time": "0001-01-01T00:00:00",
  "AnnualFrequentation": 0,
  "PanierMoyen": 0
}
```

Code Erreur Spécifique :

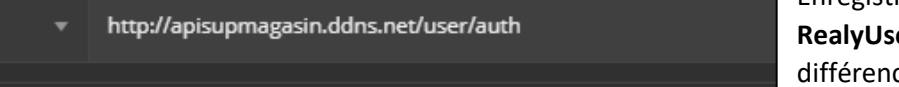
Code Erreur	Message	Raison
400	Filds Missing : X où X est le champ manquant	Champs obligatoires non remplis
550	User exist Already	Utilisateur déjà existant

Réponse :

```
{
  "login": "corentin.cannie@wanadoo.fr",
  "password": null,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1ODk5MjIwNzQsIm1zcyI6IjE1N1cF9FTWFnYXppb19FMjAyMCI6ImZyb250SW9uamMiOiJ0LgI7",
  "reallyUser": true,
  "id": "25XCC58"
}
```



L'authentification se fait par l'adresse email et le mdp enregistré en MD5 :



POST ▼ http://apisupmagasin.ddns.net/user/auth

Params Authorization Headers (9) **Body** Pre-request Script Test

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "Login" : "corentin.cannie@wanadoo.fr",
3   "Password" : "azerty38",
4   "RealyUser" : true
5 }
```

Login(string) = adresse
Enregistré
RealyUser(bool) pour
différencier les machines
vrais users

Code	Message	Raison
400	Login Null.	Le champs LOGIN est vide
400	login not resolved.	Erreur d'authentification

The screenshot shows the 'Test Results' tab in a web browser's developer tools. The status is '200 OK', the time is '97 ms', and the size is '448 B'. The response is displayed in the 'JSON' tab, showing a successful login response with a token and a confirmation that the user is real.

```
{
  "login": "corentin.cannie@wanadoo.fr",
  "password": "",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE0Oks5MjI0MjMsImZyI6IjE1IiwiaWF0IjE1ZC16ImZyb250SW9uamM1fQ.TD...",
  "reallyUser": true,
  "id": "25XCC58"
}
```

IDCustomer dans la base

[illegible]

Toutes authentifications réussis ou pas est mis dans des logs classés par jour.
Afin de repérer des éventuelles attaques ou break-force.

V) Connexion Bluetooth

Dans le cahier des charges une localisation par Bluetooth était demandée à la suite de tests et comme expliqué plus haut, le test ne fut pas concluant.

La connexion Bluetooth s'effectue en 2 temps :

- La reconnaissance Bluetooth
- L'identification au sein du magasin

a) *La reconnaissance Bluetooth*

Chaque téléphone des clients est enregistré dans la base de données dans le model phone comprenant le model du téléphone, sa marque, son numéro et surtout son adresse mac.

Cette adresse est unique pour chaque téléphone car elle permet d'identifier la puce réseau utilisé par le téléphone. Cette information est enregistrée lors de l'inscription par le biais du téléphone.

Tous les 15 secs, des bornes Bluetooth, dans notre projet des raspberry pi, scanne l'ensemble des téléphones pris dans le magasin et l'envoi sur l'api via la route (/bluetooth).

Cette route demande un mot de passe, afin de sécuriser la connexion en cas d'interception de la route par un tiers non autorisé. Le choix de cette route est dû à une facilité à changer les bornes sans pour autant l'enregistrer dans le système. Lorsque le scan est fait, il est envoyé dans sur l'api. Cette dernière l'inscrit dans un fichier temp.

Cette mise à jour est effectuée toutes les 15 secondes, d'une part par contrainte matérielle, d'autre part car 15 secs laissent le temps, en cas de défaillance de connexion de pouvoir réagir.

Code d'erreur spécifique :

Code erreur	Message	Raison
421	"Key not correspond. Are you Porcine ?"	Mauvais code de verif
420	"Phones attributs is empty"	Listes vides



b) L'identification au sein du magasin

Lors de la connexion par l'appli, web ou mobile, une demande est envoyée sur la route /bluetooth/{Mac} qui va renvoyer un boolean indiquant la présence du téléphone dans le magasin et l'ID Magasin.

Si oui, elle ouvrira les portes de la partie panier, et achat.

Sinon l'utilisateur ne pourra que consulter son compte ou son profil.

Code Erreur	Message	Raison
425	"MAC Adress not found"	Le mac présenté n'est pas dans la liste



Les Bornes Bluetooth

Nous utilisons un Raspberry pi 3 disposant d'un module Bluetooth pour la découverte des périphériques environnants.

Un package a été créé afin de procéder à l'installation et au démarrage du service de manière automatisé.

Il comprend 2 scripts, 2 fichiers texte et un fichier .exe, le package doit être placé dans les documents.

Le service se trouvera au chemin : 'etc/systemd/system/serviceBlu.service'.

Le script Deployment.sh est utilisé une fois pour l'installation du service et son lancement.

Le script scanService.sh est lancé automatiquement par le service précédemment créé.

EnvoiApi.exe sert à envoyer les données du scan une fois celui-ci terminé.

Le fichier README.txt décrit les étapes à suivre.

Le service scanne les périphériques toutes les 10 secondes et enregistre l'adresse mac et le nom des périphériques dans un fichier texte, celui-ci sera écrasé à chaque scan.

Un envoi des noms est effectué à la fin de chaque scan, si un échec de transfert arrive, le processus recommence jusqu'à 3 fois avant de s'arrêter en erreur.

Une vérification de l'adresse MAC est ensuite effectuée pour vérifier si le client est enregistré dans la base de données, si c'est le cas, un QR code lui est envoyé sur son smartphone afin de lui autoriser l'accès au magasin.

La localisation des clients via Bluetooth n'est pas fiable du tout, cette technologie n'étant pas adaptée à cette utilisation la marge d'erreur est bien trop importante.

Un système de Beacons Bluetooth localisant les clients dans le magasin n'est donc pas viable.

Une solution possible serait de réaliser du motion tracking grâce à des caméras.

Bien entendu, toutes les informations récoltées servant à suivre les clients dans le magasin seront effacées définitivement lorsque celui-ci quittera le magasin.

L'authentification via le bluetooth n'est également pas fiable, les adresses mac étant de plus en plus protégées tant par Android que Apple, le nom du périphérique et la seule autre info utilisable, mais celui-ci est facilement modifiable.

