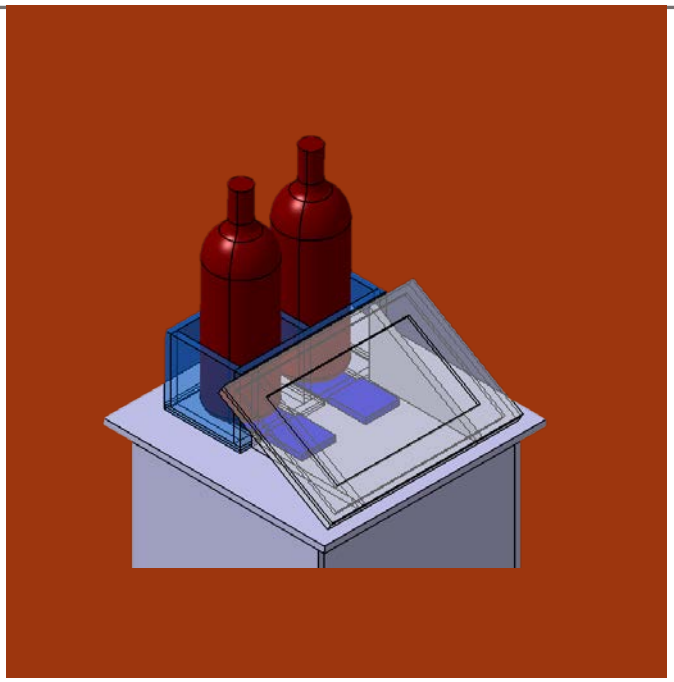


Borne œnologique

Rapport projet de fin d'étude
Février 2014

Quentin LAMBERT – Davy RIBREAU
Tuteurs : Thomas Vantroys – Yvan Peter

lifl
Université Lille1
POLYTECH LILLE

Sommaire

Table des matières

1 Introduction

Dans le cadre de notre projet de fin d'études, nous avons choisi de travailler sur la mission de création d'une borne œnologie. Cette borne sera un système informatisé permettant d'aider les clients de grandes surfaces à choisir leur vin. Ainsi, la borne agira tel un expert en vin, donnant des informations très précises sur le vin sélectionné, tout en prodiguant des conseils sur les mets à associer à ces vins.

D'un point de vue technique, afin de simplifier l'interaction avec l'utilisateur, il est prévu d'équiper les bouteilles de vins avec des tags RFID, ce qui permettra une reconnaissance aisée des bouteilles de vins par la borne. En effet la technologie RFID va remplacer le système de code barre dans quelques années.

Par ailleurs, une application à destination des clients sera créée afin qu'ils puissent emporter certaines fonctionnalités de la borne avec eux, et également afin de communiquer avec la borne.

Dans ce rapport de projet, nous examinerons dans un premier temps les différentes solutions envisagées afin de réaliser la borne, puis nous étudierons plus en détail la solution technique retenue. Enfin, nous détaillerons la création de l'application à destination des clients, avant de finir par la réalisation physique de la borne.

2 Objectifs et cahier des charges

2.1 Les objectifs

De nos jours les rayons des grandes surfaces possèdent de nombreuses références dans le domaine de l'œnologie. Les vins blancs, rosés, rouges sont pléthores. Les amateurs de vin ainsi que les clients peu avisés en la matière sont souvent à la recherche d'informations et de conseils. A l'instar des petits commerces, les grandes surfaces ne peuvent avoir un vendeur de vin conseillant les clients sur leurs choix, c'est pourquoi le développement d'une borne œnologique a été commandé. Celle-ci pourra vous conseiller et vous informer sur vos choix d'une manière simple et efficace.

2.2 Cahier des charges

Le principe de la borne œnologique est d'aider les clients de grandes surfaces à choisir leurs vins en fonction de leurs besoins et de leurs goûts.

La borne devra répondre aux fonctions suivantes :

- Obtention d'informations additionnelles concernant un vin
- Comparaison de deux bouteilles de vin
- Sélection de vins répondant à certains critères, après avoir répondu à un questionnaire
- Interaction de la borne avec le smartphone du client

3 Prospection de solution

3.1 Les bases de l'architecture

Pour les besoins de la mission, nous avons besoin de deux lecteurs RFID, afin de pouvoir scanner les tags RFID fixés aux bouteilles de vin. Nous avons décidé de relier ces deux lecteurs à un Arduino, celui-ci permettant un interfaçage de qualité avec n'importe quelle interface utilisateur.

Cet Arduino sera relié à un système plus haut niveau, qui comportera également l'interface utilisateur de la borne. Ce système haut niveau sera constitué autour d'un PC ou autour d'une tablette Android, les avantages et les inconvénients des deux plateformes seront exposés dans les prochains paragraphes.

Ainsi, l'architecture se composera des éléments suivants :

- Une tablette
- Un écran tactile
- Un Arduino
- 2 lecteurs RFID
- 4 LED Puissante
- 2 Moteurs réducteur

3.2 Architecture à base de PC

La première solution prise en compte fût une architecture à base de mini PC. Cette architecture a pour avantage une gestion de la communication avec l'Arduino relativement aisée. En effet, le programme de la borne serait réalisé en C et la liaison serait une liaison série.

Concernant l'interface utilisateur, elle serait réalisée en C# ou Java. A noter qu'il faudrait se doter d'un écran tactile. Compte tenu des coûts liés à l'obtention d'un écran tactile de bonne facture ainsi que des différents éléments à lier entre eux, nous n'avons pas retenu cette solution.

3.3 Architecture à base d'Android

La solution à base de tablette Android a pour principal avantage d'apporter un dispositif complet à bas prix, bénéficiant de technologies de communications intéressantes pour notre projet : wifi, Bluetooth, NFC.

De plus, cette solution a pour bénéfice de profiter de la qualité des tablettes actuelles. A savoir un bon comportement tactile (Multitouch) et une bonne qualité d'écran (définition de l'écran).

Cependant, la communication d'un Arduino avec une tablette Android est moins aisée que sur un PC. En effet le Java n'est pas très adapté à ce genre de communication.

3.4 Solution retenue

Après réflexion, nous avons choisi la solution à base d'Android, principalement pour bénéficier d'une base à bas prix possédant toutefois les technologies indispensables à la bonne réalisation de notre projet, comme le NFC ou le Bluetooth. Nous profiterons alors également de la qualité des tablettes actuelles, en termes d'interface. Enfin, cela nous permettra également d'approfondir nos connaissances vis-à-vis de la programmation sur Android. De même nous pensons que le futur de ces tablettes est l'interaction avec leur environnement, et en cela l'interaction entre la borne et l'Arduino (que cela soit en Wifi ou par câble) nous paraît précurseur.

Pour résumer, voici les éléments qui composeront l'architecture :

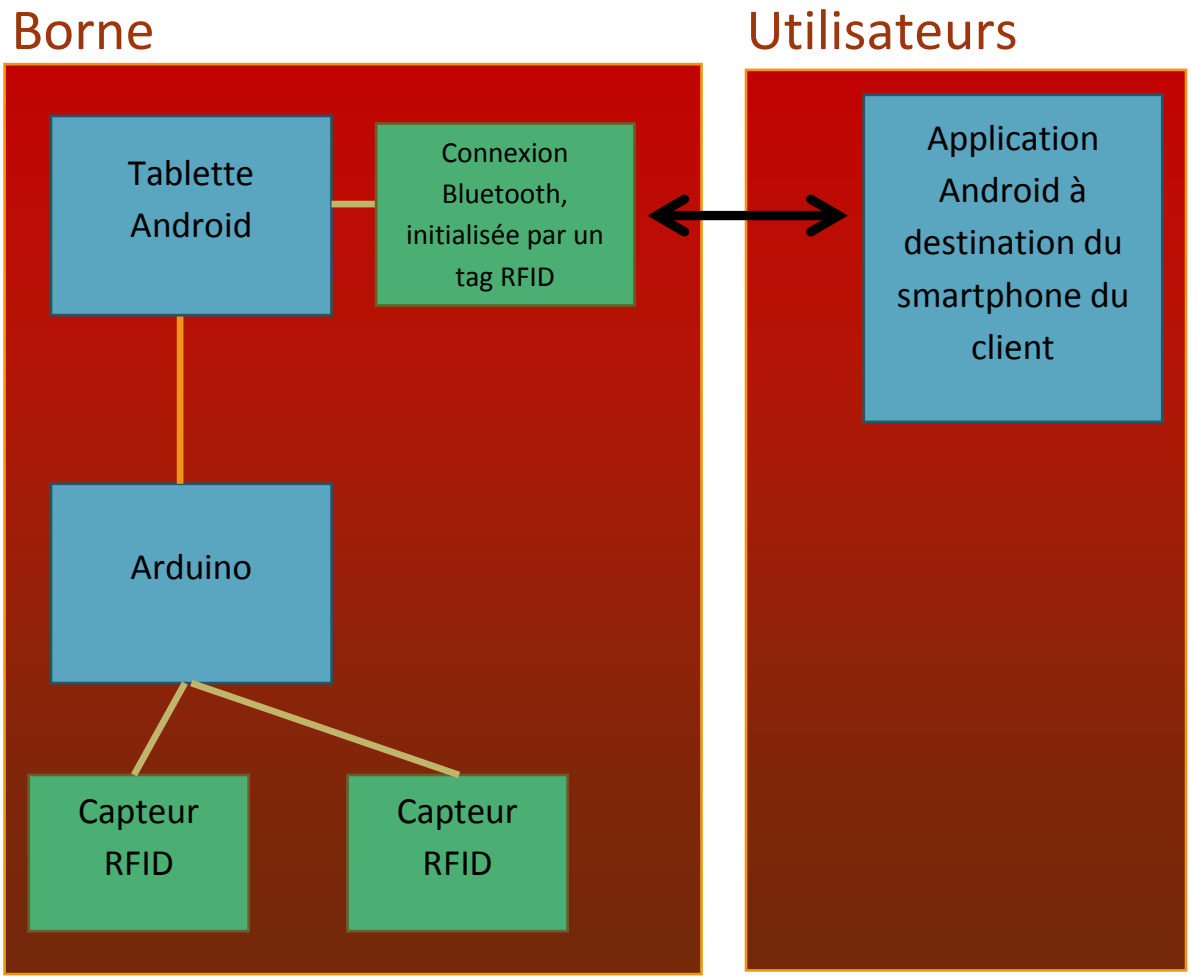
- Une tablette Android 7'' (Nexus 7)
- Une étiquette RFID, afin d'automatiser la connexion Bluetooth
- Un Arduino ADK
- 2 lecteurs RFID
- Une borne conçue par le département CM (Axel Anicet et Jonathan Mondé CM5)
- 4 LED puissantes
- 2 Moteurs réducteurs

4 Travail software réalisé

4.1 Détail de l'architecture

Afin de rendre notre projet plus attrayant aux yeux d'une entité comme une grande surface, nous avons voulu compléter notre dispositif de borne avec une application à disposition des clients. Cette application a pour objectif d'augmenter l'interaction avec le client. En effet, en plus de pouvoir accéder à certaines fonctionnalités de la borne directement depuis son smartphone, l'application permettra également de communiquer avec la borne.

Voici un schéma représentant l'architecture globale du système :

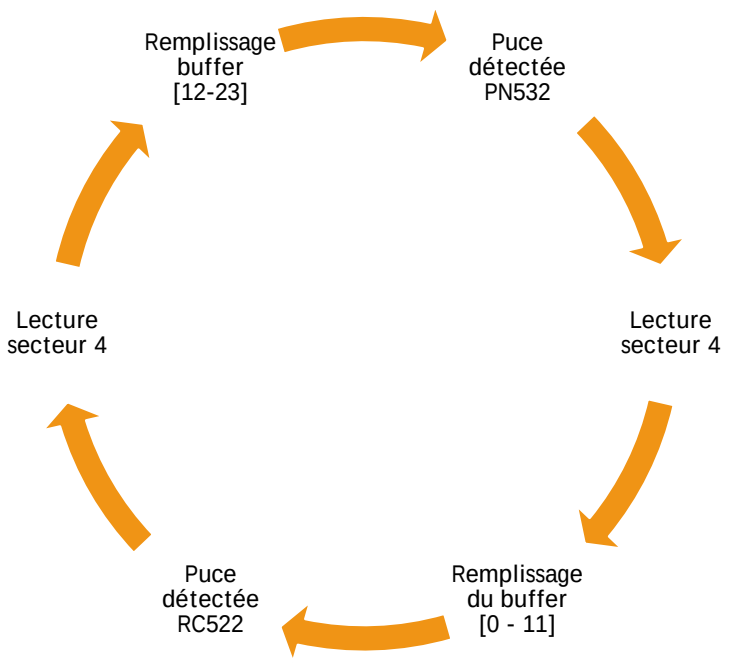


4.2 Connexion des lecteurs RFID à l'Arduino

La connexion des capteurs RFID à l'Arduino se fait via 2 protocoles. Le premier est l'I2C pour le Shield basé sur une puce PN532. Le second étant connecté en SPI basé sur une puce RC522. Bien que le bus I2C soit particulièrement adapté à ce type de montage, les puces PN532 possèdent les mêmes adresses, ce qui nous rendait impossible l'utilisation de ce protocole avec deux Shield. Le Shield Adafruit PN532 est alimenté en 5 volts tandis que le second est alimenté en 3 volts, le second est donc moins puissant et possède une portée légèrement inférieure.

Pour l'utilisation des Shield nous avons utilisé les librairies fournies par les fabricants. Toutefois, nous avons dû les modifier pour les rendre pleinement compatibles avec notre méthodologie. En effet les librairies étant bloquantes ; c'est-à-dire que tant que le Shield n'était pas en dialogue avec une puce RFID celui-ci bloquait le programme. Compte tenu de la conception des librairies, nous ne pouvions utiliser d'interruption RTI (Real Time Interrupt). Nous avons aussi rajouté une fonction aux librairies, celle d'indiquer la non-présence de puce RFID.

Le fonctionnement de la partie Arduino gérant les capteurs RFID est le suivant :



Le buffer est composé d'un tableau d'entier non signé, de 24 cases ([0-23]). A chaque tour de boucle, lorsque le capteur ne détecte aucune puce RFID alors le buffer est rempli de 12 bytes valant 0.

Prenons un exemple, nous avons 2 codes-barres RFID « 123456789101112 » et « 800851234567 », le premier sera positionné sur le capteur 1 et le second sur le capteur 2. Voici les différents cas de remplissage du buffer possible :

```
1 Bouteille sur le Shield 1          ===> 123456789101112000000000000
1 Bouteille sur le Shield 2          ===> 000000000000800851234567
1 Bouteilles sur les Shield 1 & 2    ===> 123456789101112800851234567
```

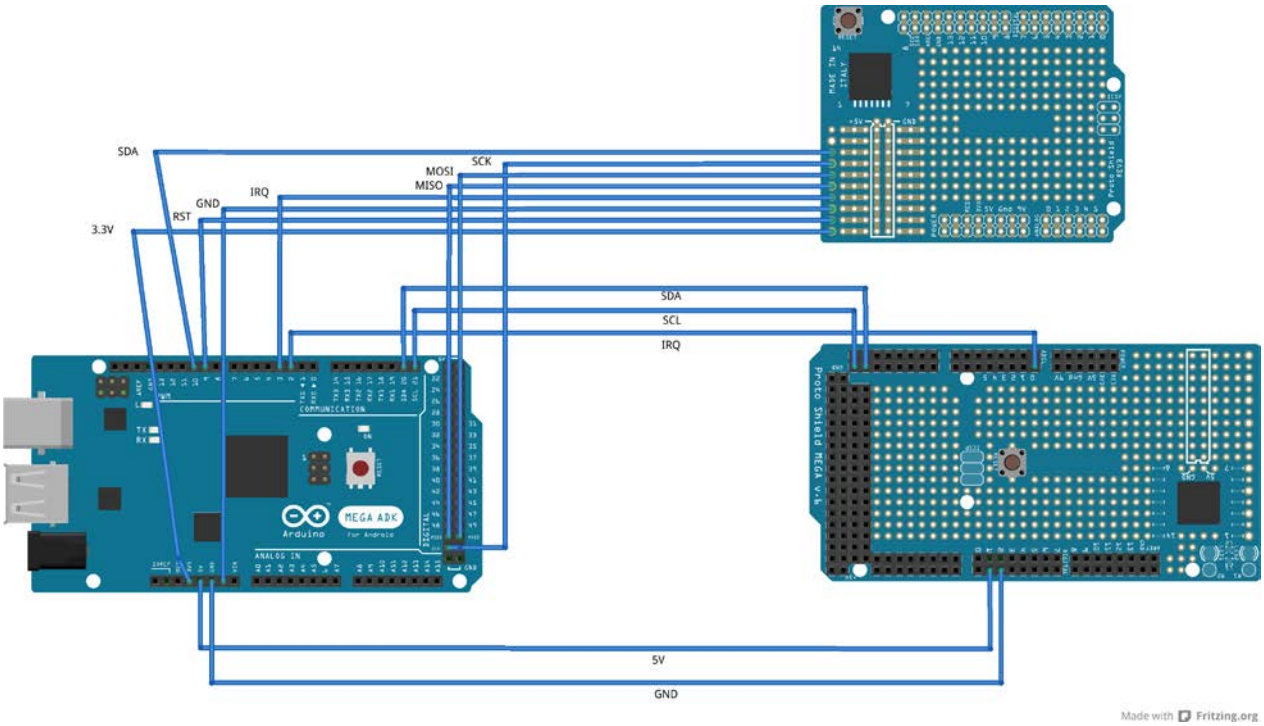


Schéma de montage des capteurs

4.3 Interfaçage de l'Arduino avec la tablette Android

Pour la gestion de la transmission d'informations entre la tablette Android et l'Arduino, nous utilisons la librairie « USB HOST SHIELD 2.0 » de *Oleg Mazurov*. Cette librairie nous permet d'utiliser le USB-HOST de l'Arduino ADK afin de faire transiter des informations (tableau de bytes) entre les 2 appareils via une liaison série. L'utilisation de l'Arduino ADK permet d'alimenter la tablette via le port HOST.

A chaque tour de boucle, l'Arduino utilise cette librairie pour transmettre le buffer à la tablette. Le procédé est assez rapide, 1 seconde suffit pour que la carte soit détectée, les données transférées et le résultat affiché sur la tablette. La transmission de l'Arduino vers la tablette s'effectue grâce à la commande suivante :

```
rcodeTX = adk.SndData( 24, CODE );
```

CODE étant le nom du buffer, et 24 le nombre de cases le composant.

La librairie est initialisée avec certains paramètres, permettant de démarrer l'application au branchement de l'Arduino sur la tablette, ou d'indiquer l'adresse du téléchargement si non présente.

```
ADK adk(&Usb,"Tisc", // Manufacturer Name
             "BornOenologique", // Model Name
             "Lancement de l'application", // Description (user-visible
string)
             "1.0", // Version
             "
https://play.google.com/store/apps/details?id=com.tisc.cmcdirect", // URL
(web page to visit if no installed apps support the accessory)
             "123456789"); // Serial Number
```

Au niveau de la réception sur la tablette côté borne, nous utilisons les fonctions d'Android 4.0 de base. C'est-à-dire lire le buffer du connecteur USB. Pour cela nous autorisons l'application Android à entrer en communication avec l'accessoire USB (l'Arduino). Les informations d'initialisation de la librairie « USB HOST SHIELD 2.0 » doivent être les mêmes que sur la tablette.

Une fois la liaison entre les deux appareils créée nous lisons le buffer, de manière cyclique. Pour cela nous créons un Handler, qui sera lancé par le système toutes les 950ms.

```
    Handler handler = new Handler();


    private final Runnable sendData = new Runnable(){
        public void run(){
            try {
                read();
                handler.postDelayed(this, 950);  ///cycle de 950ms
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
```

La fonction *read()* présente dans le fragment de code ci-dessus nous permet de récupérer les données contenues dans le buffer. Celle-ci teste la présence d'élément dans le buffer, puis récupère les données via `mInputStream.read(bufferReception)` puis les traite pour séparer les informations transmises par les 2 capteurs RFID.

```
   public void read() {
        // Lecture USB
        //*
        byte[] bufferReception = new byte[24];

        if(mInputStream != null) try {
            int status = 0;

            status = mInputStream.read(bufferReception);  ///renvoie le nombre de
byte dans "status"


            if (status > 0) {
                String v1 = "";
                String v2 = "";
                String tmpTST = "";
                v1          =          String.valueOf(bufferReception[0])          +
String.valueOf(bufferReception[1])    +    String.valueOf(bufferReception[2])    +
String.valueOf(bufferReception[3])    +    String.valueOf(bufferReception[4])    +
String.valueOf(bufferReception[5])    +    String.valueOf(bufferReception[6])    +
String.valueOf(bufferReception[7])    +    String.valueOf(bufferReception[8])    +
String.valueOf(bufferReception[9])    +   String.valueOf(bufferReception[10])    +
String.valueOf(bufferReception[11]);
                v2          =         String.valueOf(bufferReception[12])          +
String.valueOf(bufferReception[13])    +    String.valueOf(bufferReception[14])    +
String.valueOf(bufferReception[15])    +    String.valueOf(bufferReception[16])    +
String.valueOf(bufferReception[17])    +    String.valueOf(bufferReception[18])    +
String.valueOf(bufferReception[19])    +    String.valueOf(bufferReception[20])    +
String.valueOf(bufferReception[21])    +    String.valueOf(bufferReception[22])    +
String.valueOf(bufferReception[23]);



                }
        } catch (IOException e) {
            Log.e(TAG, "read failed", e);
        }

    }
```

4.4 Logiciel Arduino : Gestion des LED et Moteurs

Des LED sont connectés à l'Arduino pour rendre l'utilisation de la borne plus « Design » et accueillante. Celles-ci sont allumées par l'Arduino, via une sortie logique. Lorsque l'un des détecteurs détecte un tag RFID, la LED est allumé rendant ainsi un effet lumineux sur la bouteille.

Pour la gestion des moteurs, nous utilisons des interruptions RTI (Real Time Interrupt). Les moteurs étant des moteurs Pas-à-Pas, la commande des changements de phases doivent être précise. Pour cela nous utilisons la librairie « MsTimer2 ». Les moteurs étant à 7 phases, l'Arduino gère continuellement le changement des phases, les unes derrière les autres, la continuité de celles-ci est donc primordiale.

```
void setup()
{
  …
        MsTimer2::set(0, MOTEUR); // 0ms period
        MsTimer2::start();
  …
}
```

Initialisation des RTI

4.5 Application de la tablette Android de la borne

L'application Android de la borne regroupe l'ensemble des fonctionnalités directement disponibles depuis le magasin pour le client. Elle a pour principal objectif d'offrir au client une interface le conseillant quant au choix de son vin.

L'application propose ainsi pour chaque vin un ensemble de plats s'y associant, et parfois l'avis d'un viticulteur professionnel.

Pour cela, l'application travaille sur 4 objets correspondant à des entités différentes :

- Vin
- Type de vin
- Plat
- Avis

Techniquement, ces objets sont stockés dans une base de données SQLite. Nous apporterons plus de détails sur la façon dont les données sont stockées dans le paragraphe « 4.8 Base de données utilisée ».

4.5.1 Fonction primaire : Chargement RFID des vins

La fonction primaire de la borne Œnologique est l'affichage de données correspondant à un vin, ou alors la comparaison de 2 vins lorsque les bouteilles sont posées sur la borne.

Cette application regroupe les vues suivantes :

- Page d'accueil : elle invite le client à poser 2 bouteilles de vin sur la borne. Le client peut également accéder à l'interface de conseil de vins selon le plat depuis cette page.

- Page de comparatif : Une fois que le client a posé 2 bouteilles, cette vue se déclenche automatiquement. Elle permet de comparer les caractéristiques des deux vins posés sur la borne.
- Liste de vins : Cette vue présente la liste de vins s'associant aux plats précédemment sélectionnés. Le type de vin ainsi que son prix est directement visible depuis la liste.
- Page de présentation d'un vin : Cette vue permet de visualiser l'ensemble des caractéristiques d'un vin.

Enfin, lorsque l'application présente les caractéristiques d'un vin, ou le comparatif entre deux vins, l'utilisateur à la possibilité d'ajouter dans les favoris de son application les vins affichés sur la borne.

Pour cela, l'utilisateur doit approcher son smartphone d'un tag RFID fixé sur la borne, ce tag permet d'initialisé une connexion Bluetooth entre le smartphone du client et la borne. Ainsi, les données sont téléchargées sur le smartphone du client via la connexion Bluetooth, sans qu'aucune action ne soit nécessaire de la part du client, ce qui rend le procédé très intuitif. Le processus de connexion est détaillé dans le paragraphe « 4.7 Liaison entre la borne et le smartphone du client ».

4.5.2 Fonction secondaire : Un plat, un vin

Les clients d'une grande surface n'ont peut-être aucune idée de ce qu'ils doivent boire en fonction de leur plat. C'est pour cela que nous proposons une fonction supplémentaire. C'est n'est plus un vin ⇔ un pat mais un plat ⇔ un vin. Le client entre donc dans ce mode et est dirigé, en répondant à des « Questions » représentées par des images, à son plat. A partir de ceci, nous affichons une liste de vins correspondant au plat, triés par prix et par type (Blanc, Rosé, Rouge).

Interface de choix du plat
(De gauche à droite puis de haut en bas : Entrée, volailles, Viandes, Poissons, Fromages, Dessert)

Les deux modes de fonctionnements de la borne sont visibles en annexe.

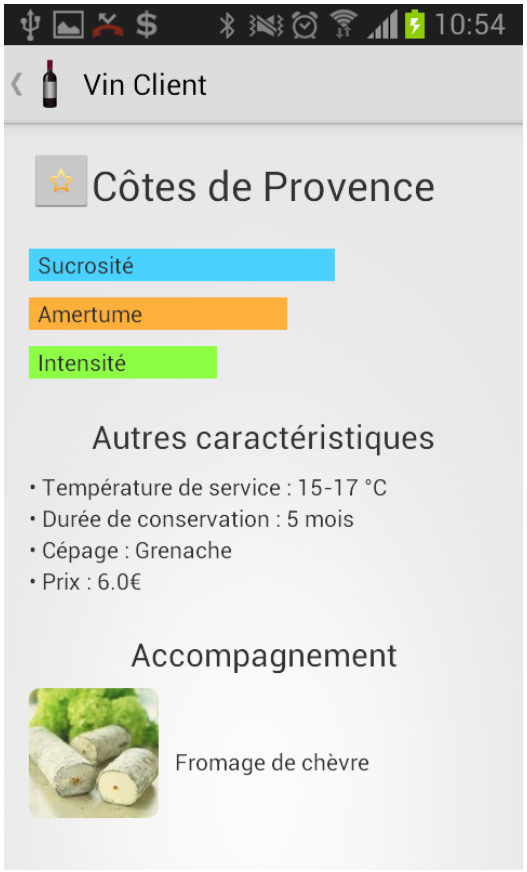
4.6 Application à destination des smartphones clients

Cette application est très similaire à l'application de la borne. Elle permet au client d'emporter avec lui la plupart des fonctionnalités de la borne.

Elle reprend notamment quelques vues de l'application de la borne :

- Page d'accueil : Cette page permet au client de sélectionner différents filtres afin de lui proposer une liste de vin répondant à ses critères
- Liste de vins : identique à l'application de la borne
- Page de présentation d'un vin : identique à l'application de la borne, à l'exception que l'utilisateur peut ajouter le vin dans ses favoris
- Liste des favoris : accessible depuis le menu de l'application, cette vue liste l'ensemble des vins mis en favoris.

La capture d'écran suivante nous présente la vue de la présentation d'un vin, on peut s'apercevoir que cette vue agrège les données de 4 objets (vin, type de vin, plat, avis) :



Les autres vues de l'application Android pour les clients sont visibles en annexe. De même, le diagramme UML des classes est disponible en annexe.

4.7 Liaison entre la borne et le smartphone du client

Afin d'ajouter une certaine interactivité entre l'utilisateur et la borne, il nous a semblé bon de créer une communication entre le smartphone du client et la borne. Cette connexion permet d'ajouter en favoris sur son smartphone les vins affichés sur la borne.

Afin de réaliser cette communication, nous avons opté dans un premier temps pour le protocole Wifi Peer to Peer, qui permet d'établir une connexion wifi entre 2 dispositifs tout en ayant de bons débits et une bonne portée.

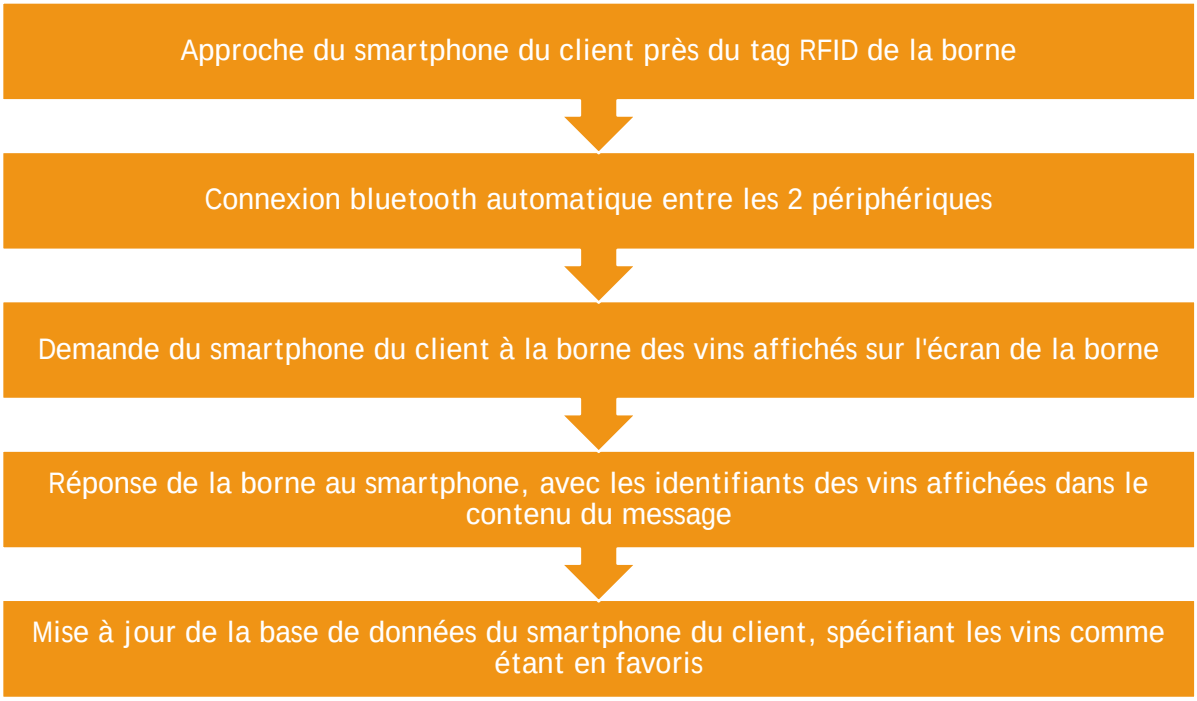
Cependant, après développement de la solution, certains inconvénients de ce type de connexion nous semblent rédhibitoires. En effet, la connexion n'est pas stable. De plus, d'un point de vue ergonomique, l'utilisateur doit activer manuellement ce type de connexion dans les paramètres avancés de son smartphone, puis autoriser manuellement la connexion, ce qui serait une véritable perte de temps pour le client.

C'est pourquoi nous avons décidé d'essayer un autre protocole : le Bluetooth. La connexion est davantage en adéquation avec ce que nous recherchons. En effet, il est possible de rendre l'établissement de la connexion entièrement automatisé sans qu'une quelconque action ne soit demandée à l'utilisateur. Ainsi il est possible d'activer le Bluetooth du téléphone, télécharger les données entre la borne et le smartphone, puis fermer la connexion, tout en rendant le processus invisible aux yeux de l'utilisateur, ce qui rend le procédé intuitif et agréable pour l'utilisateur. De plus, la portée reste correcte, plus de 10 mètres en l'absence d'obstacles.

D'un point de vue technique, un service gérant la connexion Bluetooth *« BluetoothChatService »* est lancé en tâche de fond de l'application de la borne et de celle du client. Les deux applications sont ainsi capables de communiquer entre elles à l'aide d'un protocole simple que nous avons mis au point :

- Si le message « 1a » est reçu par la borne, celle-ci doit envoyer les identifiants du/des vin(s) affiché(s)
- Les identifiants des vins sont envoyés sous la forme « vX », avec X l'identifiant du vin
- Si plusieurs vins sont envoyés à la fois, les identifiants sont séparés par des deux points « : »

Ensuite, l'application du client met à jour sa base de données afin de spécifier que les vins sont en favoris. Ils sont ainsi facilement accessibles sur l'application du client grâce à un écran dédié. Voici un schéma récapitulatif du processus :

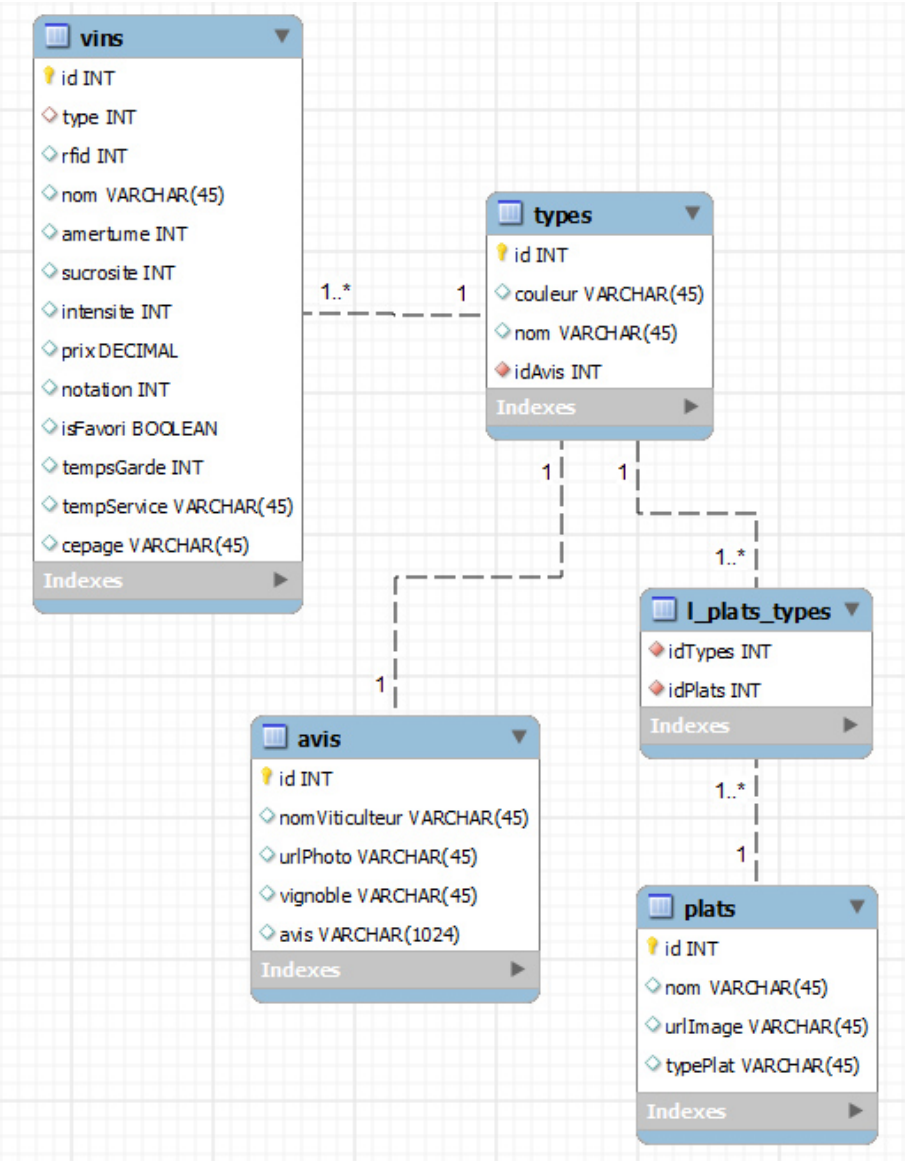


4.8 Base de données utilisée

La base de données créée est aussi bien utilisée sur l'application de la borne que pour l'application à destination du client.

Dans une première version, la base de données permettait seulement de décrire des vins et des plats, avec une table pour chacune des deux entités. Ce modèle de base de données était malheureusement trop limité par rapport aux fonctionnalités que nous souhaitions offrir. Nous avons donc revisité l'ensemble de la base de données afin qu'elle soit plus flexible et plus fonctionnelle.

Au final, notre base de données permet de décrire trois types d'objets : les vins, les types de vin, les avis associés, et les plats. Concernant la cardinalité, un à plusieurs vins peuvent appartenir à un type de vin. On associe à chaque type de vin un avis et une liste de plats. Ceci nous donne le schéma UML suivant :



Vu globale de la base de données

A noter qu'étant donné la cardinalité entre la table "type" et la table "plat", une table de liaison nommée "l_plats_types" fait le lien entre les données de la tables "types" et la table "plats".

D'un point de vue technique, à chaque entité représentée (qui peuvent être du type vin, type de vin, plat, avis), une classe de DAO a été créée, permettant de manipuler la base de données pour l'entité donnée. DAO signifie "Data Access Object", il s'agit d'un design pattern permettant de s'abstraire de la façon dont les données sont stockées. De même, chaque entité est associée à une classe Java, créée afin de la décrire, avec un ensemble de champs privés et d'accesseurs public. Par exemple si l'on veut récupérer les données concernant un plat, nous ferons appel à une méthode de la classe DAOPlat qui nous renverra directement un objet Plat, dont les caractéristiques seront accessibles par un ensemble de méthode public.

4.9 Peuplement de la base de données

Un important travail a été fourni afin de peupler convenablement la base de données. Ce travail concerne d'une part la collecte d'informations concernant les vins, mais également la technique qui a été employée afin de peupler aisément cette base de données.

En effet la méthode de peuplement est importante pour rendre la tâche de peuplement le moins fastidieux possible, mais également pour éditer des mises à jour de la base de données facilement.

Nous avons pour cela décidé d'utiliser un logiciel nous offrant une interface graphique afin d'ajouter des entrées, ce logiciel se nomme " SQLite Database Browser" (visible en annexe). Ce logiciel nous fournit en sortie une base de données au format SQLite. En complément de cela, nous utilisons dans notre application Android une librairie tierce permettant d'importer au démarrage de l'application notre fichier de base de données. Cette librairie se nomme "Android SQLiteAssetHelper".

5 Réalisation de la borne

5.1 Plans initiaux

Initialement, nous avions prévu de réaliser la borne entièrement en plexiglass. Nous avions fait appel à Axel Anicet (étudiant CM5) afin de concevoir les plans. Il s'est malheureusement avéré qu'en plus du coup élevé de la matière première, l'atelier mécanique de l'école était incapable d'usiner les pièces. Nous avons donc dû nous retourner vers une autre solution, tout en gardant notre ligne de conception aiguillée par Axel Anicet.

5.2 Assemblage de la borne

Après prospections de différentes solutions, nous avons donc choisi de réaliser nous-même la borne en bois.

Pour cela, le LIFR nous a fournis différents matériaux restants d'anciens projets :

- 2 planches de bois 120*60*1,5 cm
- 1 planche de bois 120*60*0,5 cm
- 1 plaque de plexiglass 100*50*0,5 cm
- Peintures, pinceaux, etc.

Etant donné le changement de situation, les plans créés ont dû subir quelques modifications, surtout au niveau de l'emplacement des différents capteurs, afin que nous puissions plus facilement usiner la borne. Nous avons ainsi découpé les planches nous-mêmes à l'aide de l'atelier bois de Polytech.

5.3 Mise en rotation des bouteilles

Les tags RFID équipant les bouteilles ne se trouvant que sur une surface limitée des bouteilles, nous avons mis au point un système permettant de mettre en rotation les bouteilles tant qu'elles n'ont pas été détectées. Ceci permet à l'utilisateur de poser les bouteilles dans n'importe quelle orientation, tout en permettant que les bouteilles soient scannées avec un taux de réussite maximum.

Ce dispositif est identique pour les deux bouteilles, il est constitué d'un plateau tournant en plexiglass sur lequel est posée la bouteille. Ce plateau est attaché à une tige filetée sur laquelle un engrenage est fixé. Tout ceci est ensuite mis en rotation à l'aide d'un moteur pas à pas équipé d'un engrenage sur son pignon. Aux yeux de l'utilisateur, seul le plateau tournant est visible, le reste se situe sous la plaque de bois principale.

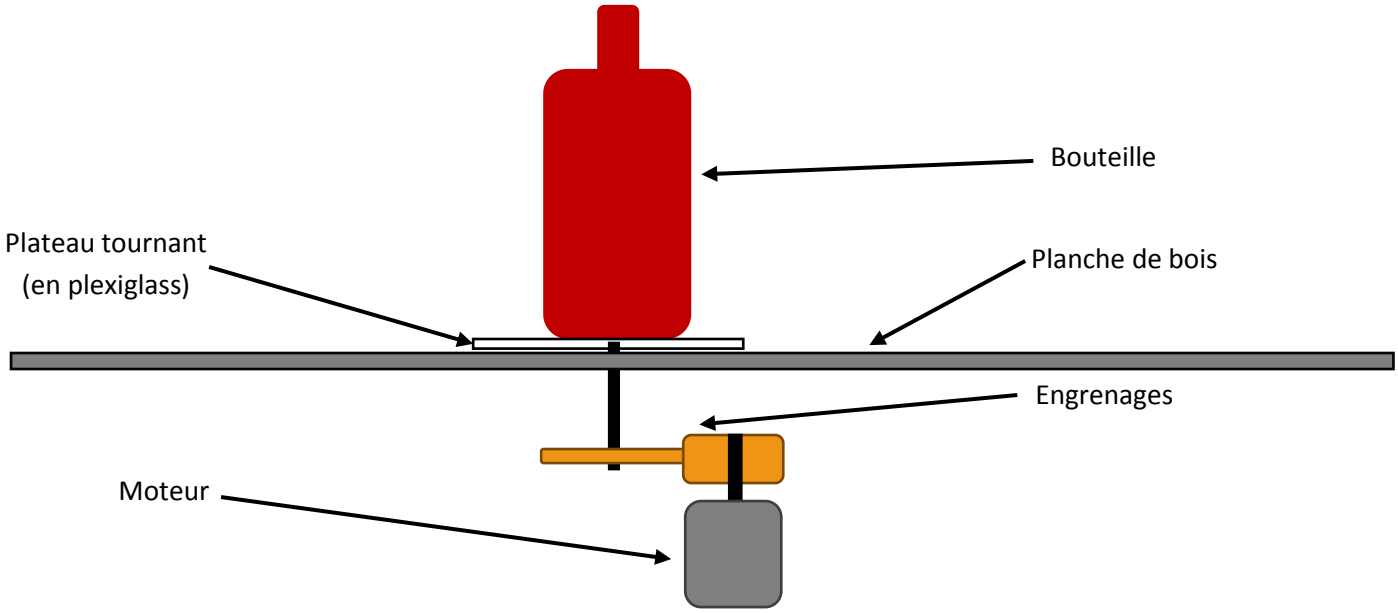


Schéma du dispositif de rotation d'une bouteille

5.4 Alimentation de la Borne et du système

L'alimentation de la borne se fait via le secteur, un transformateur est intégré pour transformer du 220 volts en 9 volts afin de pouvoir alimenter l'ensemble des composants. L'Arduino n'étant pas assez puissant pour alimenter l'ensemble : moteurs, tablette Android, LED, et récepteur RFID cumulant plus de 700mA en consommation (maximum délivré par l'Arduino) nous avons mis en place une carte de routage de l'alimentation. Celle-ci permet d'alimenter l'Arduino via un plug de 3.5mm ainsi que des header (x6) pour les moteurs pas à pas à alimentation externe comprise entre 5-12v. Les LED sont quant à elles connectées sur la carte de routage pour les mêmes raisons, c'est-à-dire une bonne répartition de l'énergie.

Un plug supplémentaire a été prévu, pour un éventuel rajout futur de composants.

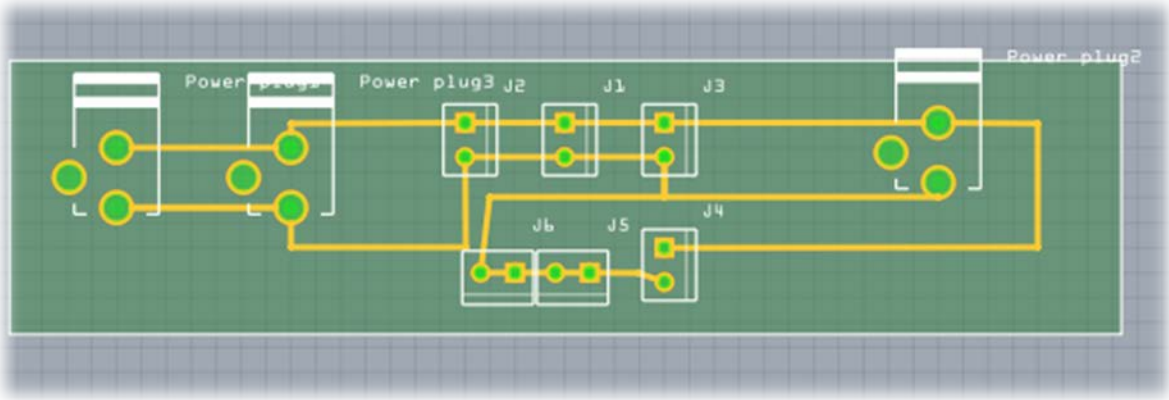


Schéma de la carte de routage de l'alimentation de la borne

6 Conclusion

Ce projet nous a permis de toucher à l'ensemble des compétences attendues par un ingénieur IMA. En effet, nous avons dans un premier temps développé une solution software sur Android, dont l'une des deux applications développées est interfacée avec un microcontrôleur Arduino. Le code Arduino que nous avons développé permet quant à lui de gérer les ordres bas niveaux tels que l'allumage des LED et de moteurs, mais également de récupérer les valeurs des tags RFID.

Enfin, nous avons conçu et réalisé la borne, afin que l'ensemble du système soit fonctionnel et attractif pour l'utilisateur en mettant à profit une collaboration avec des étudiants du pôle mécanique (Axel Anicet et Jonathan Mondé) pour la conception de la borne, ce qui permet un échange de compétences entre les secteurs, ce que l'on peut retrouver dans les entreprises.

7 Annexe

Vue de la borne en utilisation

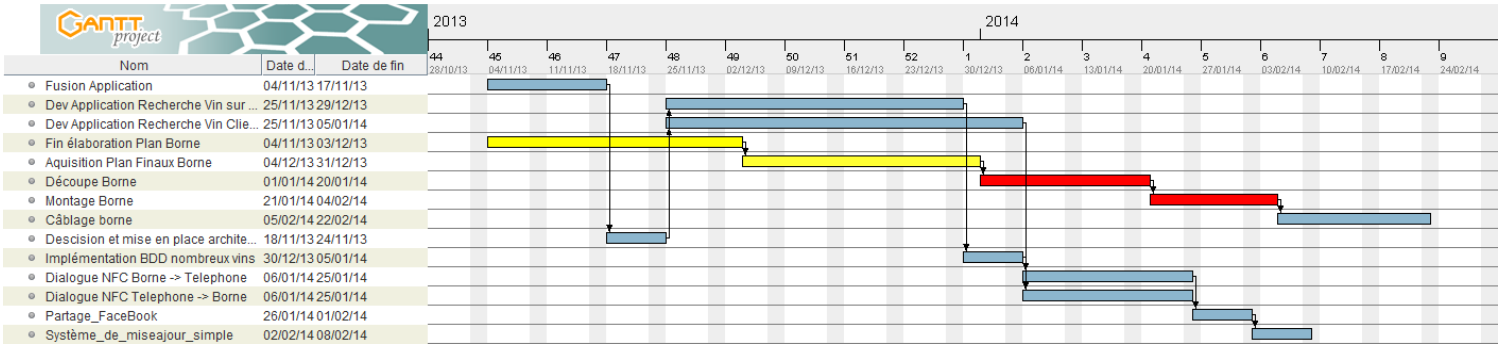
Borne vue de face

Plat	Vin / Appelation / Nom	Type
	APERITIF	
--	Sancerre	Blanc Sec
--	Pouilly Fumé	Blanc Fruité
--	Pouilly Fuissé	Blanc Fruité
	ENTREE	
Foie gras	Coteau du Layon, Sauternes, blanquette de Limoux	Blanc Liquoreux
Asperge à la Flammande	Côte du Rhône, Saint chinian	Rouge
Brochette de Jambon au melon	Bandol, Beaujolais	Blanc leger Sec
Coquille Saint-Jacque	Ladoix	Blanc Sec fort
	Poisson et fruit de mer	
Poisson cru et fumé	sauvignons de Touraine, chablis, Rieslings, Gewürztraminer	Blanc Fruité
Poisson cuisiné	Côtes de Provence, Beaune, Meursault, Savennières, Chassagne-Montrache	Blanc Sec
Filet de rouget	Côtes de Provence	Blanc Sec
	FROMAGE	
Mozzarella et fromages frais	Bourgogne-aligoté, Mâcon, Chablis, Saumur	
Roquefort et fromages bleus	Sauternes, Coteaux-du-layon, Montbazillac, Bordeaux, Banyuls	Blanc Liquoreux
Brie, Camembert	Sauvignon, Sancerre, Pinot blanc	Blanc Fruité
Fromage de chèvre	Sancerre, Chablis, Muscadet, Côte de Provence	Blanc Sec / Blanc Fruité
Beaufort, Gruyère, Comté	Givry, Chardonnay, Mâcon, Château-Chalon	Blanc Sec
Saint-Nectaire, Saint-Paulin	Bourgogne	Rouge
Munster, Pont-l'évèque, Epoisses	Gewurztraminer, AOC Alsace-Muscat, Chablis, Pinot blanc	Blanc Fruité
	DESSERT	
Fruit Rouge	Champagne Rosé Demi-Sec	Demi-Sec
Chocolat	Banyuls, Maury	Rouge Liquoreux
Glace	Liqueur	Liqueur
Caramel	Monbazillac, Jurançon	Blanc Liquoreux
Café	Banyuls, Maury, Sauterne	Rouge Liquoreux
A base d'épice	Macvin du Jura	Blanc Liquoreux
Fruit	Sauternes, barsac, bonnezeaux (Anjou), champagne demi-sec, gewurztraminer vendanges tardives	Blancs Liquoreux

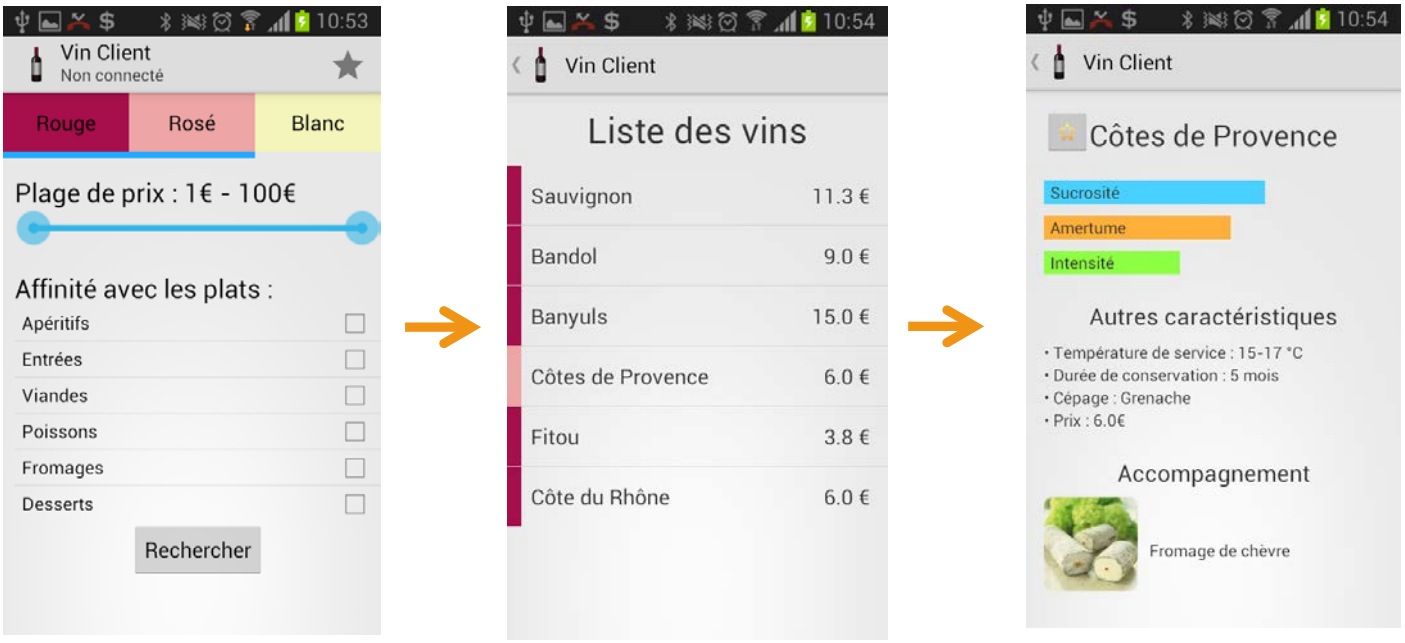
Exemple de correspondances de plats, et de vins

Appelation /Nom	Type	Cépage	Garde	Sucrosité	Température de Service	Prix
Beaujolais	Blanc leger Sec	Chardonnay	2 Ans	--	10 c°	7 €
Bandol	Blanc leger Sec	Clairette	1 à 3 ans	--	10 et 12 c°	8 €
Bandol	Rouge	--	3 à 5 ans	--	16 et 18 c°	9 €
Ladoix	Blanc Sec fort	Chardonnay	3 à 5 ans	--	10 et 12 c°	10 €
Banyuls	Rouge Liquoreux	Grenache Noir	2 à 20 ans		10 et 14 c°	7 €
Sancerre	Blanc Sec	Sauvignon	3 ans	--	10 c°	8 €
Chablis	Blanc fruité	Chardonnay	2 ans	--	9 c°	7 €

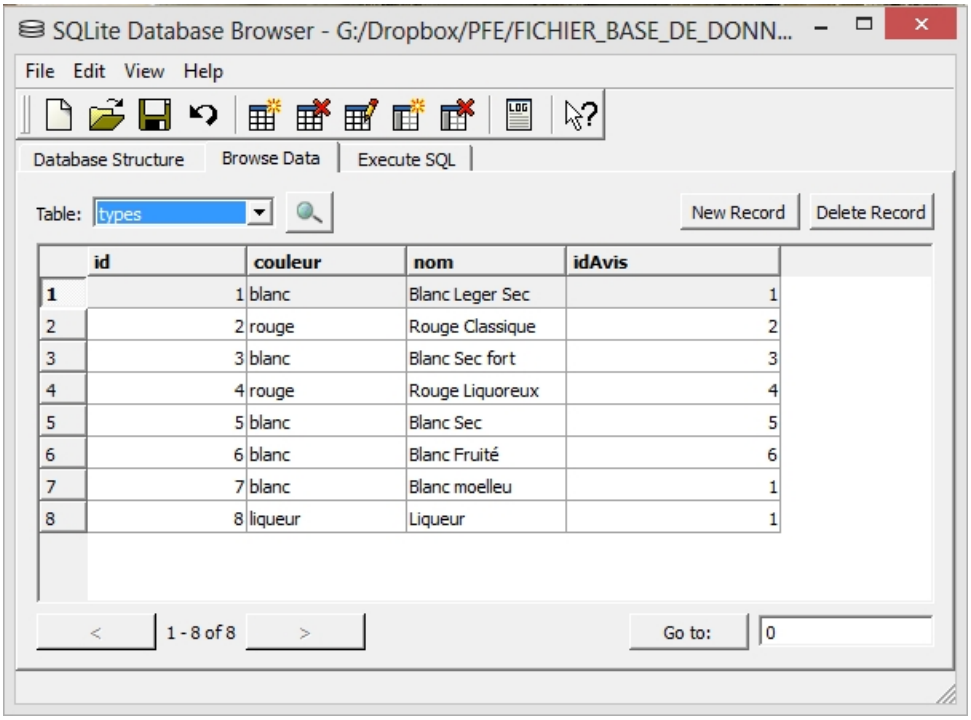
Exemple de vins intégrés à la base



Gantt montrant l'évolution du projet jusqu'à la fin, celui-ci ayant été respecté



Présentation de l'interface de l'application Android pour clients

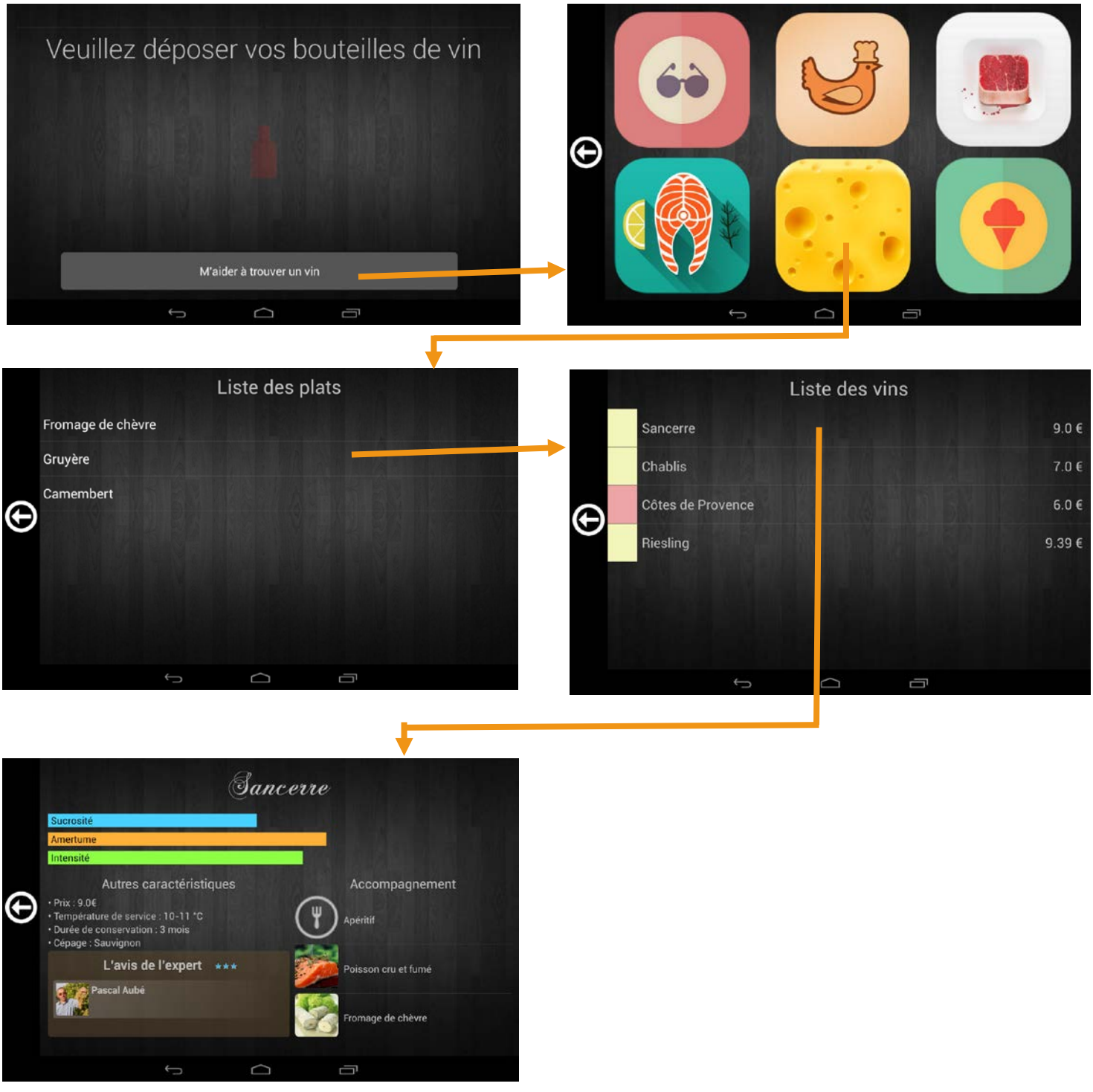


SQLite Database Browser – Ce logiciel a été utilisé pour peupler aisément la base de données.

Pose d'une première bouteille

Pose d'une seconde bouteille

Premier mode de fonctionnement de la borne



Second mode de fonctionnement de la borne

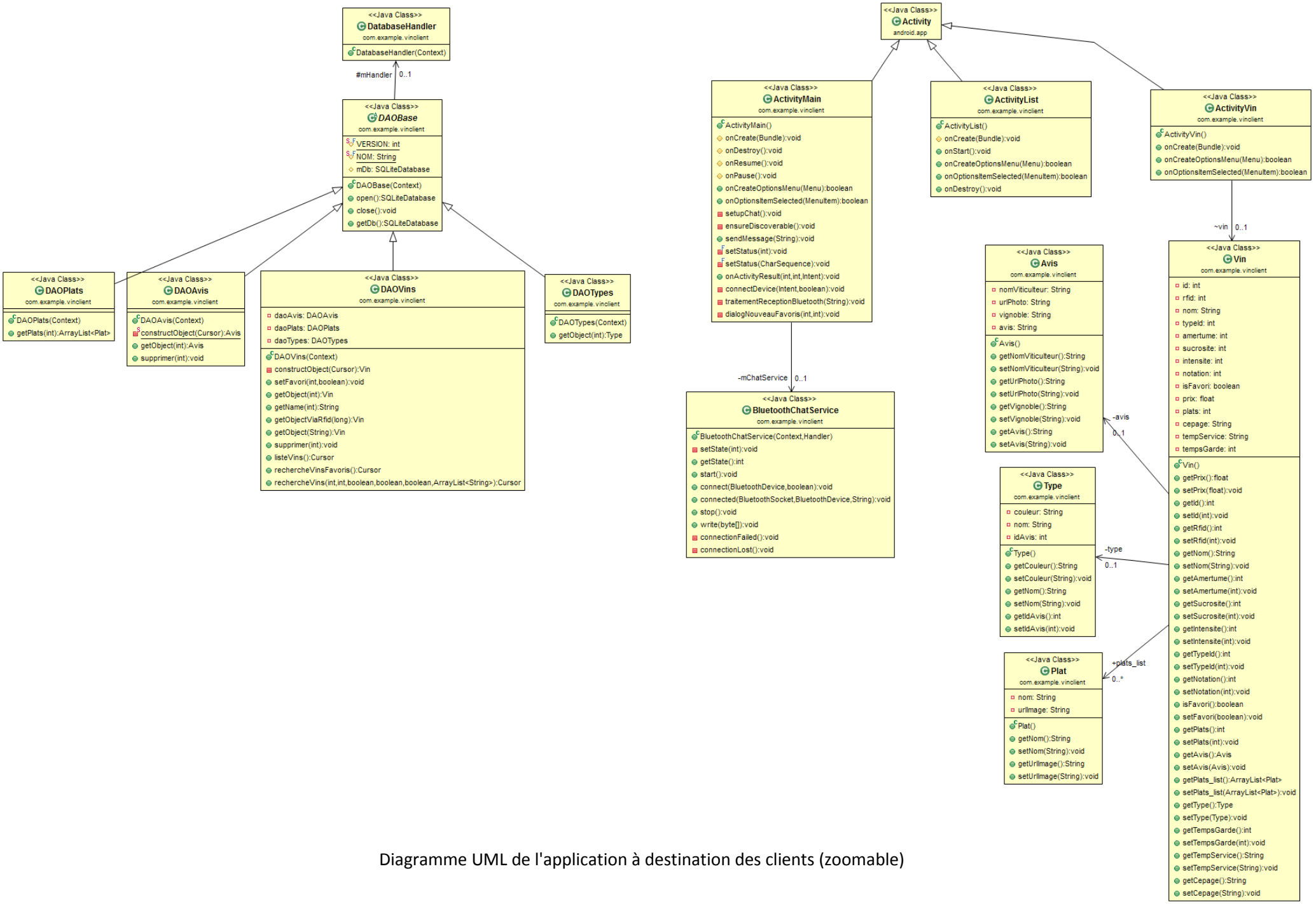


Diagramme UML de l'application à destination des clients (zoomable)