**Rapport de Stage**



*Mission : Développement d’une application Android*

FSA IUT de Valence

6, Rue Paul Henri Charles Spaak 51, rue Barthélémy de Laffemas

26000 Valence 26000 Valence

Département Informatique

**Rapport de Stage**



Maitre de stage : Jean-François COUPAT

Tuteur de stage : Sébastien JEAN

Remerciements

Je tiens à remercier toutes les personnes qui ont rendu possible ce stage. Tout d’abord, je remercie M. Jean-François COUPAT, mon maitre de stage qui m’a défini précisément les besoins de l’application et qui m’a suivi tout au long du stage.

Je remercie également Laurent DELOULME, le directeur des opérations France qui m’a admis en tant que stagiaire au sein de FSA, ainsi que Guy BONTEMPS et qui m’a donné des informations clés concernant la société. Mickael VARES m’a grandement aidé pour créer toutes les images des modes de marche. Je le remercie beaucoup pour sa sympathie et sa disponibilité tout au long de mon stage.

Je tiens aussi à remercier tous les employés de FSA Valence et tout particulièrement M. REBOUL pour l’accueil qui m’a été réservé et l’excellente ambiance qui a régnée tout au long du stage.

Je remercie enfin mon tuteur de stage M. JEAN qui a toujours suivi mon stage avec intérêt et bienveillance.

Ces dix semaines m’ont apporté énormément sur le plan professionnel évidemment mais aussi sur le plan humain, sur la gestion de projet, sur les transactions entre clients et vendeur mais aussi sur le fonctionnement d’une entreprise à l’internationale avec toutes les assistances, les réunions et les déplacements que cela implique.

Introduction

Pour conclure la formation de DUT informatique, j’ai été amené à faire un stage du 10 avril au 16 juin 2017. Ayant déjà eu l’occasion de développer sous Android durant mon projet tutoré, j’ai cherché un stage qui pourrait me permettre de continuer dans cette voie. C’est ainsi que j’ai choisi le sujet de stage de « Fabricom Systèmes d’Assemblage » (FSA) qui proposait d’adapter une serveur Web de suivi de production en une application mobile. Ce qui a pour but de pouvoir accéder de manière rapide, simple et ergonomique au suivi de la production depuis un simple smartphone.

Fabricom Systèmes d’Assemblage assemble et créer des chaines de production pour différents clients comme PSA , Renault… Devenue une entreprise multinationale, elle a des filiales en Roumanie, et, au Portugal par exemple. Du fait de la délocalisation de la production, le site de Valence est maintenant dédié à l’analyse, à la conception des projets et au service après-vente. Le seul endroit en France ou la production est toujours assurée est le site de Besançon. En effet, c’est ici que les solutions sont testés et évalués avant d’être déployés à l’étranger.

Chez FSA, chaque commande est unique et nécessite une demande précise du client qui donnera lieu à une étude de projet. De plus le faible nombre de client et le coté très concurrentiel du marché pousse FSA à proposer des solutions innovantes pour remporter les marchés. « Melodie Net » est donc un argument de poids dans les négociations. C’est pourquoi une application mobile basée sur « Melodie Net » permettrait de rendre l’outil plus complet, plus facile à utiliser et plus encore, plus moderne et intégrée dans la mode actuelle.

Encadré par Monsieur Jean-François COUPAT, responsable de l’outil « Melodie », nous avons dû définir ensemble les fonctionnalités attendues, mais également faire un choix quant aux technologies utilisées.

Basée sur le serveur web « Melodie Net », l’application Android que j’ai développé, a une importance fondamentale car elle est vouée à être utilisée dans de nombreux endroits à travers le monde. C’est pourquoi diverses contraintes ont été imposées et notamment la gestion de la langue. Toutes ces contraintes seront explicitées plus tard dans le rapport.

J’ai travaillé en collaboration avec Monsieur COUPAT. Je me suis chargé de la partie « Client », c’est-à-dire l’application Android et M. COUPAT de la partie « Serveur », c’est-à-dire le service Web sur lesquels les informations indispensables à l’application sont accessibles.

Alors quels ont été les besoins et les choix technologiques pour cette application et quels sont les étapes qui m’ont permis de réaliser une application mobile répondant à ces contraintes ?

Table des matières

[Remerciements 3](#_Toc483820033)

[Introduction 4](#_Toc483820034)

[Table des matières 5](#_Toc483820035)

[I) Présentation et objectifs généraux du projet 7](#_Toc483820036)

[A) Présentation de la société 7](#_Toc483820037)

[B) Contexte du projet et besoins du client 11](#_Toc483820038)

[C) Les besoins fonctionnels 12](#_Toc483820039)

[D) Conclusion 14](#_Toc483820040)

[II) Conception et mise en oeuvre de la solution 15](#_Toc483820041)

[A) L’élaboration du cahier des charges (modèle théorique de lappli) 15](#_Toc483820042)

[B) Les outils technologiques choisis 15](#_Toc483820043)

[C) Fonctionnement de l’application 16](#_Toc483820044)

[16](#_Toc483820045)

[18](#_Toc483820046)

[D) Les différentes phases de développement 19](#_Toc483820047)

[E) Les difficultés rencontrées et leurs solutions 19](#_Toc483820048)

[F) Conclusion 21](#_Toc483820049)

[III) Bilan technique et humain 21](#_Toc483820050)

[A) Nouvelles compétences 21](#_Toc483820051)

[B) Relations humaines 22](#_Toc483820052)

[C) Description de l’état final de l’application 22](#_Toc483820053)

[D) Perspectives futures de l’application 22](#_Toc483820054)

[E) Conclusion 22](#_Toc483820055)

[Conclusion 22](#_Toc483820056)

[Annexes 24](#_Toc483820057)

[1.1) Quelques exemples de produits fabriqués par les lignes de production FSA 24](#_Toc483820058)

[1.2) Clients de FSA 26](#_Toc483820059)

[1.3) L’application web « Melodie Net » 27](#_Toc483820060)

[1.4) Les diagrammes de l’application 28](#_Toc483820061)

[1.5) Les IHM de l’application 30](#_Toc483820062)

1. Présentation et objectifs généraux du projet
2. Présentation de la société



Chiffre d’Affaire : 22 606 800 €

**Historique :**

-La société Fabricom Systèmes d’assemblage a vu le jour en 1999 à l’issue du rachat de la société belge « Fabricom », crée en 1995, par GDF Suez.

-Par la suite, elle prend de l’importance en rachetant d’autres sociétés comme TEHNOMAT en Roumanie par exemple.

-En 2001, c’est la création de la coentreprise « JVS FABRICOM » à Sao Paulo au Brésil, en partenariat avec la société JVS EQUIPAMENTOS PARA AUTOMACAOINDUSTRIAL LTDA.

-En 2013, FSA s’internationalise encore plus avec la création de FSA ASSEMBLY INDIA PRIVATE LIMITED à Pune en Inde.

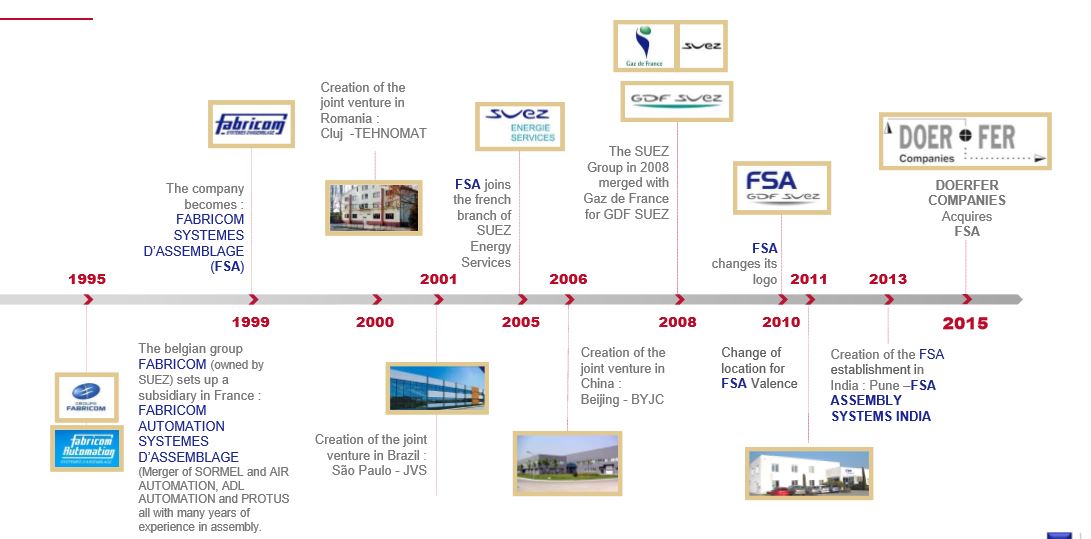


Figure 1 : Historique de la société

FSA (Fabricom Systèmes d’Assemblage) est une société au rayonnement mondial rachetée il y a peu par l’entreprise américaine DOERFER.

**Organisation au sein de l’entreprise :**

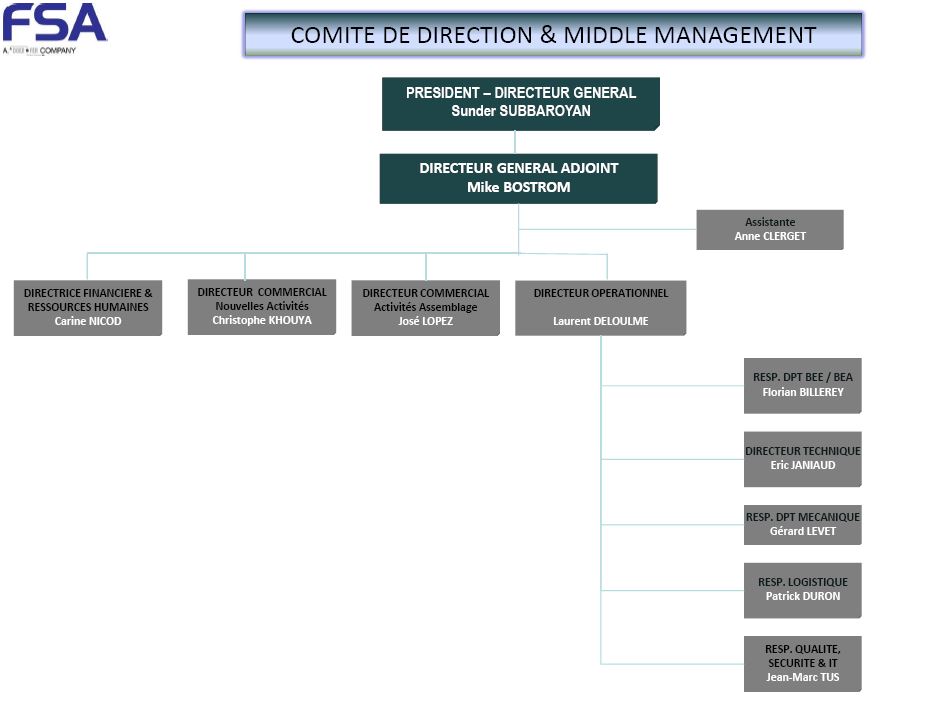
****

Figure 2 : Organigramme

**Marché et organisation :**

FSA a plusieurs particularités liées à son marché notamment. En effet, elle est capable de mener des projets de front partout sur la planète, ce qui est rare et très recherché par ses clients.

L’anglais est donc devenu un prérequis indispensable pour tous les membres de la société. Voici une illustration représentant l’implantation de FSA dans le monde :

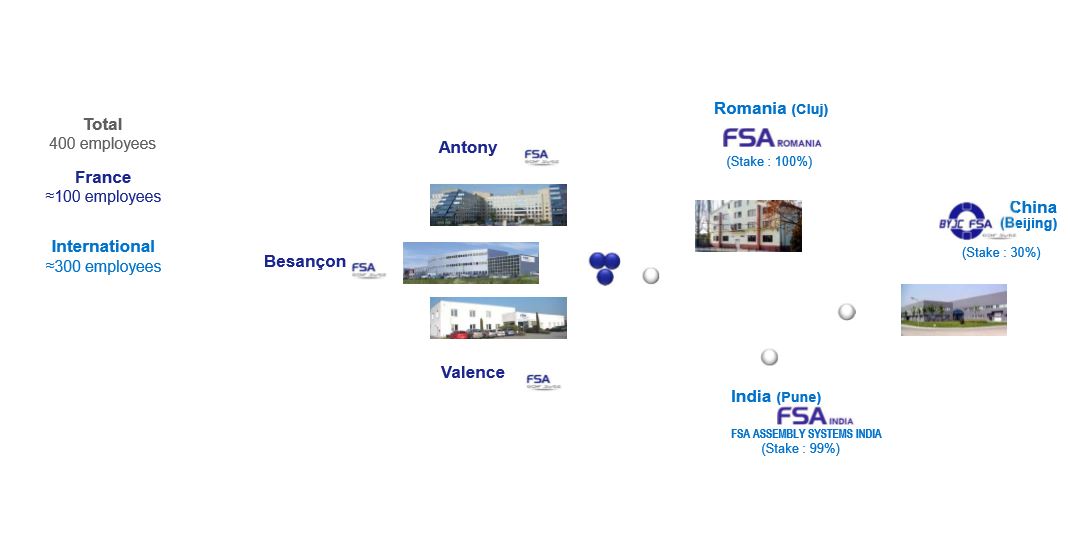


Figure 3 : Implantation dans le monde

La vocation de la société est de concevoir et installer des chaines de montage principalement dans l’automobile. Historiquement, le site de Valence est né du rapprochement avec les équipementiers automobiles de la région et le site de Besançon de la proximité avec les grands constructeurs automobiles comme PSA.

FSA fonctionne par projets, c’est-à-dire qu’elle fonctionne en fonction des projets qui lui sont proposés. Ce fonctionnement lui permet d’adapter ses moyens au projet en cours, de constituer des équipes de projets constituée de chefs de projets, d’automaticiens et d’électroniciens. Avant d’en arriver là, chaque projet passe dans une phase de pré-étude ou FSA décide si elle peut s’occuper du projet et propose un chiffrage au client. Les besoins doivent être clairs et précis car chaque écart par rapport au descriptif de l’offre donne lieu à un avenant, souvent très cher.

**Commercial :**

* 50 million d’euros de revenu

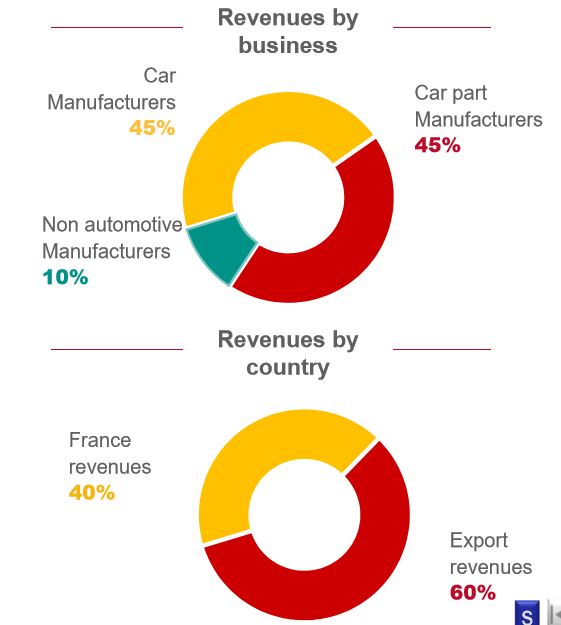


Figure 4 : répartition de la clientèle

FSA peut également être amené à fabriquer des produits quand on en trouve pas dans le commerce comme des presses électriques dans les années 1990, par exemple.

FSA a peu de clients mais il s’agit toujours de gros clients comme par exemple PSA, VALEO, RENAULT. D’où la concurrence très rude pour obtenir les marchés.

Vous pouvoir voir en annexe 1.1 quelques exemples de produits fabriqués par les lignes de Fabricom Systèmes d’Assemblage et quelques clients de FSA en annexe 1.2.

1. Contexte du projet et besoins du client

L’application « MelodieNet » permet de suivre les lignes de production en temps réel. Développée par FSA, cette application web donne accès à de multiples interfaces dans lesquels la performance de production est détaillée par des chiffres et des Pareto.

Mon but durant ce stage est d’adapter les deux fonctionnalités essentielles de « MelodieNet » à savoir le « suivi machine » et le « suivi de production ». L’un correspondant au « mode de marches des cellules » sur « MelodieNet », l’autre au « bilan de production », lequel sera légèrement simplifié. (Voir annexe 1.1). L’objectif final étant de rendre l’état des celulles d’une ligne de production choisie visualisable sur l’application. Le second objectif, classé par ordre d’importance par mon maitre de stage est de pouvoir visualiser l’état de la production pour chaque produit classé par référence. A savoir, la quantité de produits et le nombre de produits défectueux. Le dernier objectif est la visualisation dynamique...

FSA étant une entreprise très active à l’international, l’une des premières contraintes pour l’application a été de pouvoir gérer plusieurs langues, au minimum le français et l’anglais.

De plus, elle doit permettre l’authentification d’utilisateurs en vérifiant les informations entrées dans la base de données.

Toutes les informations affichées par l’application doivent être récupérées sur un Web Service dédié dont l’URL est modifiable à travers un écran de paramétrage de l’application.

Ce Web Service est de type « REST » et il exporte des données au format JSON.

1. Les besoins fonctionnels

L’une des contraintes sur l’application a été de maximiser sa simplicité d’utilisation, mais aussi de présenter les informations de manière lisible.

L’application devait impérativement être internationalisable. En effet le rayonnement international de Fabricom Systèmes d’Assemblage et plus précisément de DOERFER font de la gestion de la langue un paramètre fondamental au sein de l’application. Un smartphone étant un objet avant tout personnel, la langue de l’application sera la même que la langue du téléphone sur lequel elle s’exécute.

Au lancement de l’application, nous devrons être dirigé vers un écran d’authentification lequel nous permettra par la suite d’accéder au menu. L’enregistrement d’un utilisateur ne sera pas effectué car le client juge qu’il n’est pas nécessaire pour l’application mobile.

La base de l’application devra être un menu nous permettant de choisir entre trois écrans différents, à savoir le **« Suivi machines »**, le **« Suivi de la production »** et les **paramètres**.

Dans l’écran **« Suivi machine »,** il apparait important d’afficher le **numéro de cellule**, le **libellé de cellule** et enfin le code couleur nous permettant de connaitre **l’état de chaque cellule de la ligne**. Toutes ces informations seront présentées sous la forme d’un tableau que l’on pourra faire défiler si la place pour afficher les informations venait à manquer. De plus, l’affichage doit être « dynamique », c’est-à-dire qu’il doit permettre d’afficher les informations en temps réel par un simple rafraichissement.

Pour ce qui est de l’écran **« Suivi de production »,** il doit suivre le même principe que précédemment pour le **« Suivi machines»,** mais en affichant cette fois la référence de chaque produit mise en relation avec la quantités de produits corrects et la quantité de produits défectueux.

Dans les deux cas, le mobile devra envoyer une requête au service Web avec des paramètres qui permettront au service Web de répondre avec les informations souhaitées.

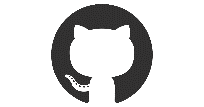
Enfin l’écran de **paramétrage** doit permettre, dans un premier temps, de modifier l’URL du service web dans l’application. Cette modification doit être sauvegardée dans l’application de manière à être la nouvelle URL même si on quitte l’application.

* **Charges d’utilisation :** 
  + S’authentifier
  + Paramétrage éventuel de l’application
  + Visualisation des informations concernant la production et les organes de production
  + Rafraichissement des informations par « scrolling »
* **Charges du système :**
  + Charges d’utilisation
  + Simplicité d’utilisation
  + Se connecter au service Web pour afficher les informations sur machines et produits en temps réel.
  + Vérifier l’identifiant et le mot de passe en se connectant à la base de données

1. Conclusion

1. Conception et mise en oeuvre de la solution
2. L’élaboration du cahier des charges (modèle théorique de lappli)
3. Les outils technologiques choisis

 J’ai choisi d’utiliser L’IDE « **Android Studio »** pour développer l’application. Il s’agit de l’environnement de développement le plus courant et le plus performant sur Android devant le plugin « ADT » de l’IDE « Eclipse » devenu dépassé par la rapidité et la simplicité d’utilisation de « Android Studio ».

Au cours du développement, j’ai utilisé « **GitHub »** associé au logiciel de gestion de version **TortoiseGit.** Le premier est le logiciel de gestion de version le plus couramment utilisé, le second permet une utilisation efficace et simplifiée de « GitHub » sans passer par des lignes de commandes sur le terminal. « TortoiseGit » est utilisé par le personnel informatique de l’entreprise

 J’ai aussi été amené à utiliser « **Gimp »** pour réaliser toutes les images des boutons de couleurs permettant à l’utilisateur de voir l’état des modes de marche dans le « suivi machines ».

 Dans la première version de l’application, avant la création du service web, l’authentification sur l’application se faisait directement entre l’application Android et la base de données. J’ai donc du prendre en main « **Microsoft SQL Server »** et de le configurer de manière à pouvoir communiquer avec la base de données grâce à des requêtes SQL. Cette communication est gérée sous Android par la librairie « **JTDS 1.2.8 »**. Le schéma explicatif est disponible dans les nouvelles compétences de la troisième partie.

La communication avec le service web a été tout d’abord gérée par la librairie externe « **Android Asynchronous http Client** » écrite par James SMITH . Nous nous sommes réunis M.COUPAT et moi afin de choisir le meilleur moyen de communication et il s’est avéré après comparatif que nous avons décidé après les avoir comparés que la technologie « **HttpURLConnection »** integré à Android était un moyen plus performant et surtout plus léger à l’utilisation. C’est donc cette solution que nous avons choisi pour la version finale de l’application.

Nous avons effectué un comparatif similaire pour déterminer le format d’échange des données entre le service web et l’application mobile. Nous avons opté pour le format « **JSON** » au détriment du format « **XML** » car il était plus léger et rendait la communication plus rapide d’après de nombreux forums.

Utilisée pour traduire les données du format « **JSON** » en chaines de caractères, la librairie « **GSON** » constitue également l’un des éléments clés de l’application.

Le lien github du projet est le suivant : **https://github.com/mordohaw/MelodieAndroid.git**

1. Fonctionnement de l’application

Il existe à ma connaissance deux moyens de communiquer avec un service Web sous Android :

**Android Asynchronous Http Client**

## 

**HttpURLConnection**

Ici, j’ai choisi d’utiliser « HttpURLConnection » car c’est le moyen le plus léger et le moins consommateur de ressources. De plus, ce moyen est directement inclus dans Android, il n’y a donc pas besoin d’ajouter des librairies externes. L’application devant être disponible sur des supports Android de version 4.4 ou plus, ce moyen est donc utilisable.

La performance prévaut sur une compatibilité poussée sur toutes les anciennes versions d’Android d’où le choix de « HttpURLConnection » au détriment de la librairie « Android Asynchronous Http Client ».

Android n’autorise pas plus de 5 secondes de traitement et donc de blocage de l’interface utilisateur. C’est pour cela que quelque soit la technologie de communication utilisée pour communiquer avec le service Web, il faut encapsuler le traitement dans une AsyncTask. Il s’agit d’un thread indépendant qui va ici lire et écrire sur le Service Web en ne bloquant donc pas l’interface utilisateur.

n obtiendra donc un schéma de ce type :

Envoi une requête

Service Web

réponse

lance

Affichage des informations

Le thread UI continue à réaliser ses traitements

Accède aux informations

Service Web

Ensuite, on a le fonctionnement suivant :

Retourne les données sous forme de chaine

Crée et donne l’Url pour accéder à l’information

Envoie la langue du téléphone

## 

hérite

Demande une Url

Renvoie l’Url correspondant

1. Les différentes phases de développement de l’application

Voici les différentes étapes du développement de l’application MelodieNet\_Android\_Client :



Pour une version plus globale, veuillez-vous reporter le diagramme de Gantt en Annexe « 1.4) Le diagramme de Gantt du projet ».

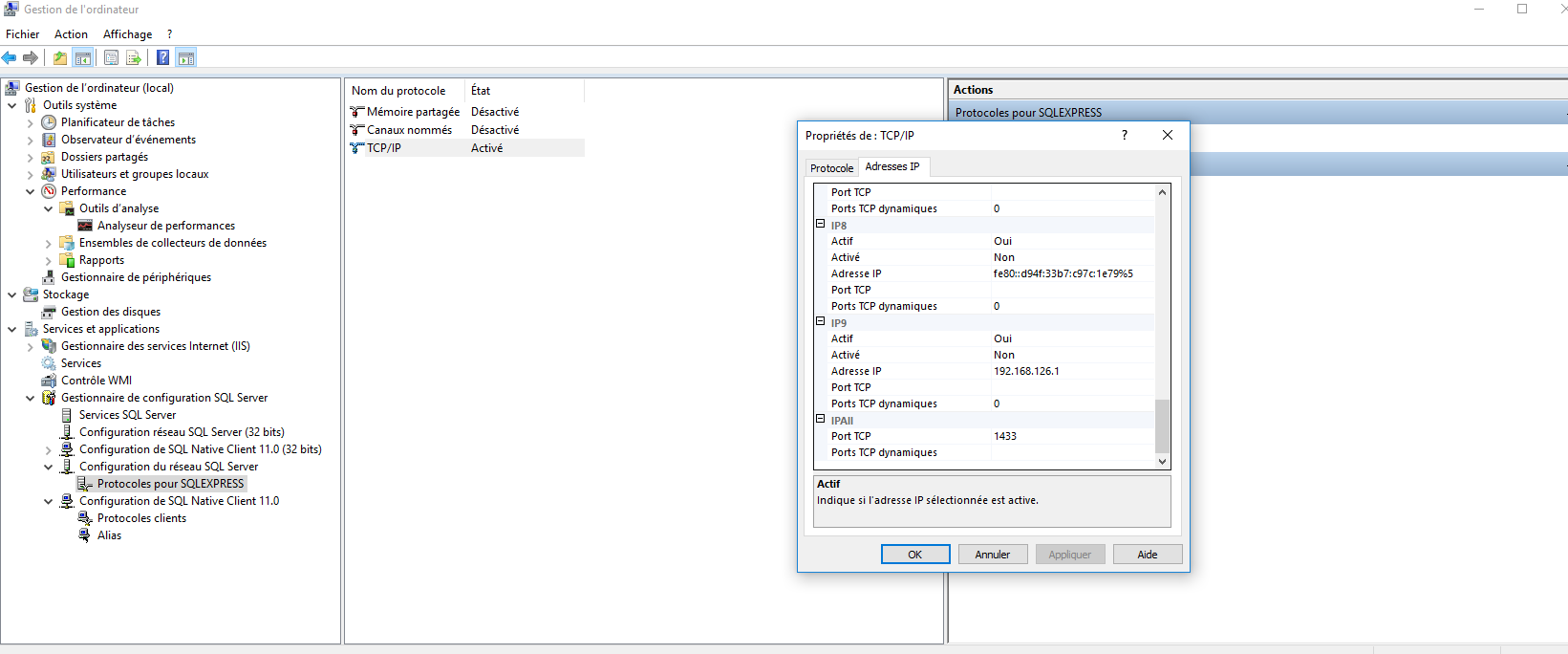
Voici ci-dessous un diagramme représentant les différentes réunions prises avec le client au cours du stage.

1. Les difficultés rencontrées et leurs solutions

* Pour appliquer un traitement à chaque élément d’une liste, il faut penser à effectuer l’opération dans l’adapteur de la liste et non pas dans l’activité qui appelle l’adapteur.
* Préremplir le champ « mot de passe » avec une valeur par défaut (type « hint ») posait problème pour les « edit text » de type « password ». En effet, dans le fichier xml, la modification de cet attribut qui fonctionnait très bien avec le nom d’utilisateur ne fonctionnait pas. Cet attribut est fondamental puisqu’il permet à l’utilisateur d’identifier les informations à entrer dans les champs. En le faisant en Java via la méthode « setHint » cela a résolu le problème.
* Ma première « réelle » difficulté a eu lieu le 27 avril. A ce moment-là, l’application Android communiquait directement avec la base de données pour vérifier les informations entrées par l’utilisateur lors de son authentification. La connexion avec la base de données échouait systématiquement pour une raison inconnue.

Après plusieurs recherches sur internet, j’ai trouvé la solution : il fallait reconfigurer les ports alloués à Microsoft SQL Server.

Pour cela, il m’a fallu aller dans « Gestion de l’ordinateur » puis aller dans la configuration du réseau « SQL server » puis protocole pour SQLEXPRESS, TCP/IP et mettre à « null » la valeur de ports TCP dynamique et mettre 1433 dans la valeur port TCP (=statique). En effet cette configuration m’a permis de me connecter à la Base de données ce qui plantait avant. Voici une illustration du moyen utilisé pour résoudre le problème :



* Le 2 mai, une nouvelle difficulté est apparue, lors de mon travail sur le hashage du mot de passe par le protocole MD5 pour la connexion. Le problème est que le mot de passe entré par l’utilisateur que je hash via la méthode hashPassword de la classe Hash ne le crypte pas exactement comme il l’est dans la Base de données.. On a « c2fca0234de447157701d0c2d64094 » au lieu de « c2fca02304de447157701d00c2d64094 » pour l’utilisateur « administrateur+ ». J’ai donc vérifié via http://md5encryption.com/ lequel était le bon. C’était celui de la base de données. Le souci provenait du mot de passe crypté : il fallait utiliser un « BigInteger » plutôt qu’un « StringBuffer ».

Le remplacement de ces lignes :

*StringBuffer sb = new StringBuffer();*

*for(byte b1 : byteTable)*

*{*

*sb.append(Integer.toHexString(b1 & 0xff).toString());*

*}*

par cette ligne :

*//Converts message digest value in base 16 (hex)*

*md5 = new BigInteger(1, digest.digest()).toString(16);*

a permis de résoudre le problème.

* Problème : j’utilisais informUser = R.string.Password\_failure; pour récupérer une chaine dans les ressources au sein du code java, ce qui déclenchait une erreur car R.string.Password\_failure est considéré comme un entier par java. J’ai donc plutôt utilisé

informUser =getResources().getString(R.string.Password\_failure);

ce qui fait bien ce que je désirais, c’est-à-dire mettre le string « Password\_failure » dans informUser.

* Pas possible de se connecter (.connect) au WebService car il faut l’exécuter en tache de fond. Android le bloque par défaut pour ne pas qu’il bloque le thread UI
* Obtenir le string résultat en sortie d’une AsyncTask : 3 paramètres : paramètres, progression et résultat. Il faut donc tout d’abord mettre le résultat comme étant un String puis on fera un .get() à la suite de l’exécution de l’AsyncTask pour obtenir son résultat.

1. Conclusion
2. Bilan technique et humain
3. Nouvelles compétences

**Authentification par accès direct à la Base de données**

Envoi d’une requête SQL

Application Android

Envoi du résultat sous la forme d’un ResultTest

BDD

Melodie

Jtds-1.2.8 library

Importation

1. Relations humaines
2. Description de l’état final de l’application
3. Perspectives futures de l’application
4. Conclusion

Conclusion

Ce projet a été divisé en plusieurs phases : la conception de l'application, le développement au cours desquels nous avons pris des réunions avec notre client et notre tuteur pour identifier nos taches.

Nous avons également gérer les coûts du projet comme dans le cas d'un projet en entreprise. La répartition des taches via Gantt Project a été décisive dans la gestion de notre temps. Elle nous a permis de maximiser le nombre de fonctionnalités de l'application mais aussi certaines fois de se concentrer sur l'essentiel, les fonctionnalités les plus importantes au détriment d'autres fonctionnalités secondaires et plus gourmandes en temps.

Le projet nous a fait aborder de multiples aspects de la programmation Android : Sur les

images tout d'abord, on nous avons réussi à utiliser l'appareil photo de la tablette pour prendre des photos que l'on a pu afficher dans l'application et modifier avec des calques par la suite.

Ces photos sont des composantes d'un constat. On peut en ajouter au cours de la création ou une fois le constat crée. De plus, nous avons pu apprendre à créer des formulaires complexes, à enregistrer les informations entrées et à les récupérer. Nous avons aussi appris à utiliser le design pattern « proxy » pour la gestion des droits d’accès lors de la connexion sur l’application.

Les perspectives d’évolution de l’application pourraient être l’ajout d’une fonctionnalité permettant de modifier un constat, de compléter un constat pré-rempli. Elles pourraient être également de récupérer les informations concernant l’œuvre

Enfin, pour conclure, nous avons beaucoup appris grâce à ce projet.

La découverte des activités du musée qui avait été l'une des raisons de notre choix, n'a pas été une déception, elle a été enrichissante pour tout le monde. Les expositions qu'il propose sont variées et les mettre en œuvre nécessite une organisation importante entre les deux partis que notre application vise, justement, à simplifier.

D'autre part, modéliser les besoins d'un client, lui poser les bonnes questions pour cerner le cadre de l'application a été une phase fondamentale dans le projet puisqu'elle nous a permis de mettre en application beaucoup de cours afin d'obtenir les informations utiles pour notre projet. Ces informations nous ont permis par la suite de produire la solution la plus proche et la plus adaptée possible aux besoins du client.

De plus, nous avons tous pu acquérir de solides compétences sur la programmation Android et les outils de développement associés. Chacun a des compétences différentes correspondant à son rôle dans le développement mais tout le monde a acquis cette base commune qui lui sera sans doute d'une grande utilité dans des projets Android futurs.

Annexes

* 1. Quelques exemples de produits fabriqués par les lignes de production FSA

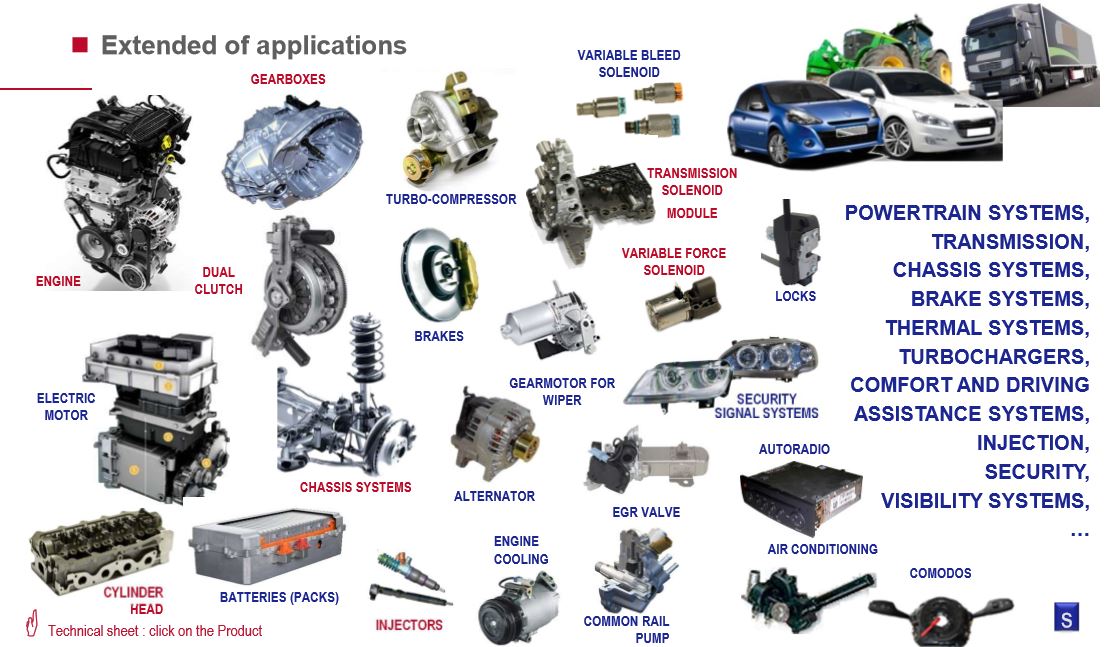


Figure 5 : Produits pour l'automobile



Figure 6 : Produits pour d'autres industries

* 1. Clients de FSA



Figure 7 : Clients

* 1. L’application web « Melodie Net »



Figure 8 Suivi des machines

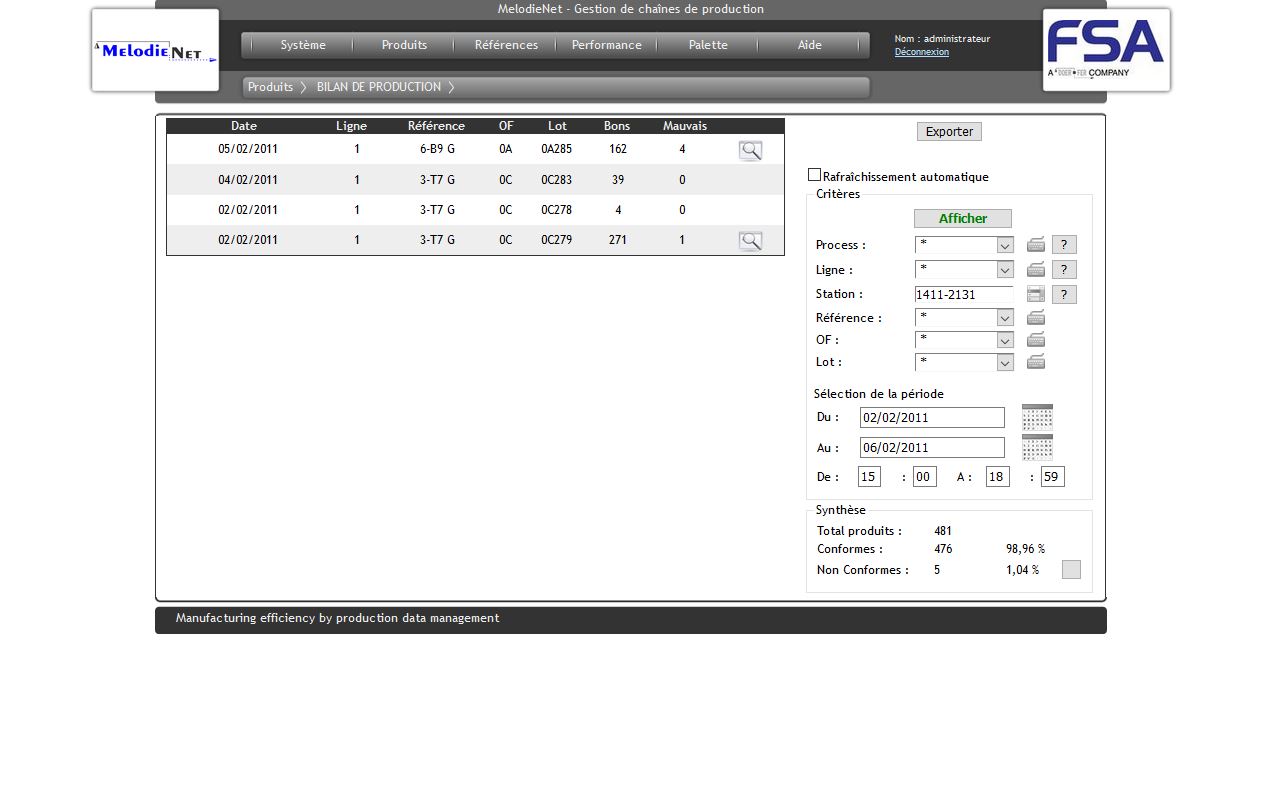


Figure 9 Suivi de la production

* 1. Le diagramme de Gantt du projet

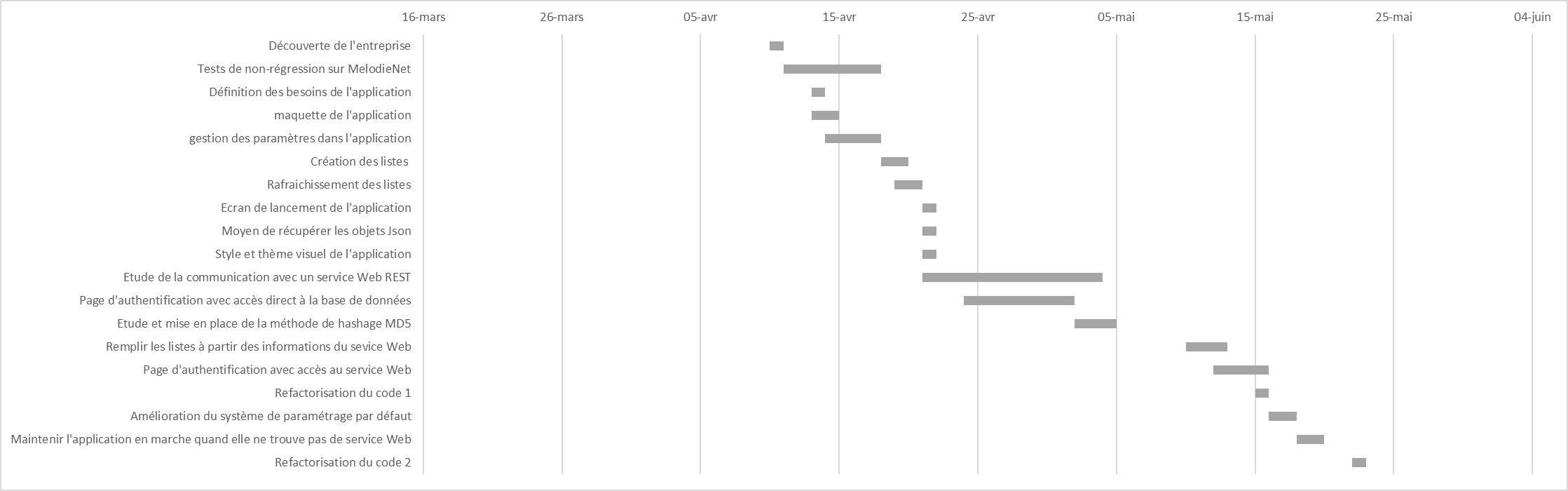


Figure 10 : diagramme de Gantt

* 1. Les diagrammes de l’application

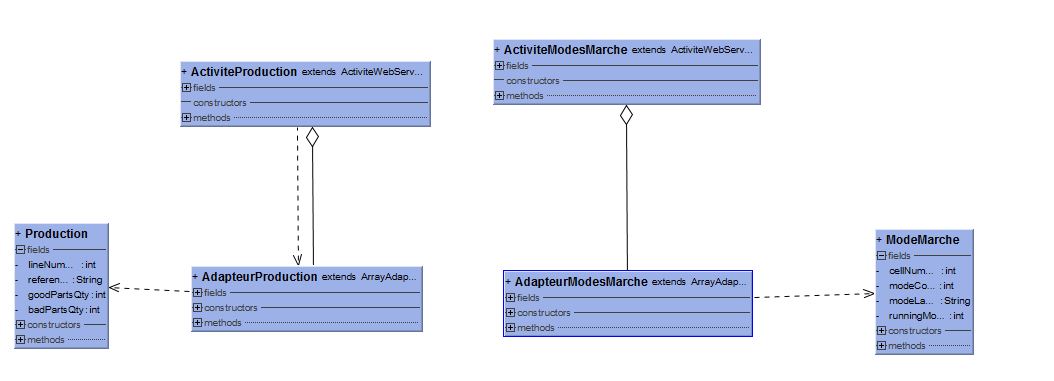


Figure 11 : diagramme de classe pour les listes

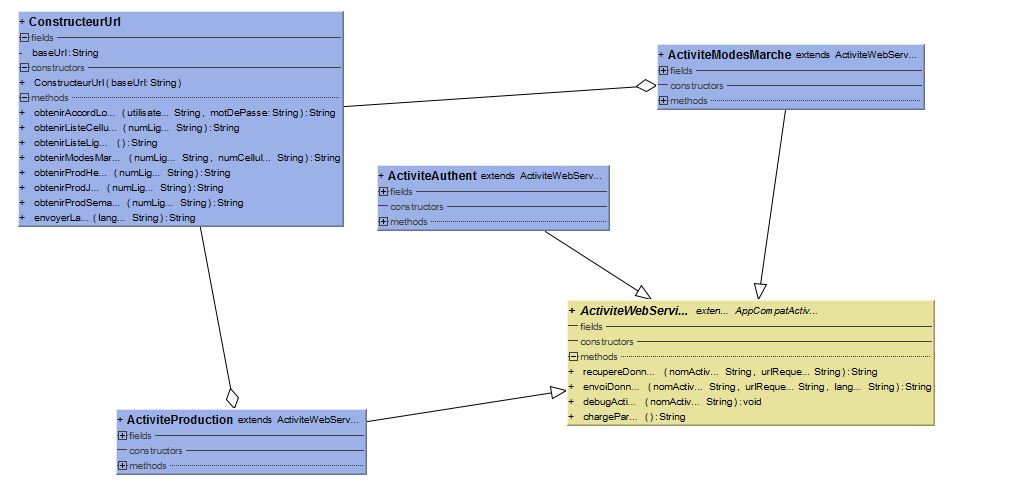


Figure 12 : diagramme de classe de communication avec le service Web

* 1. Les IHM de l’application

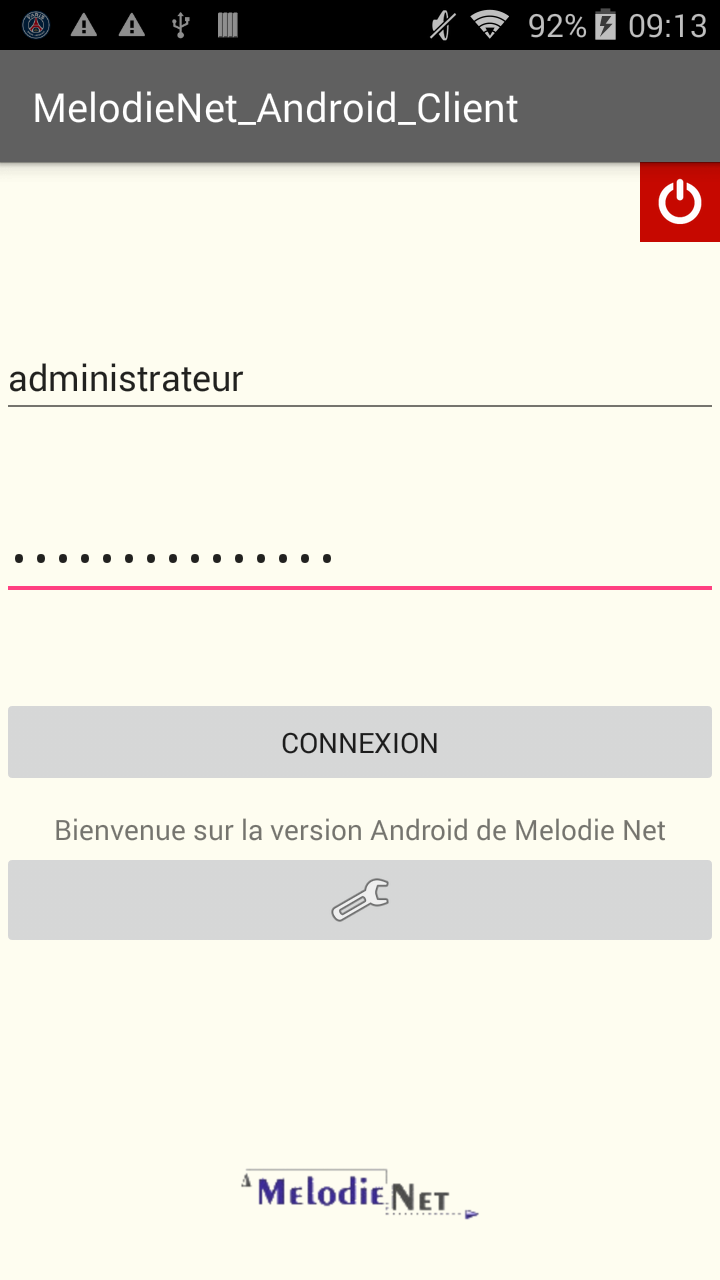
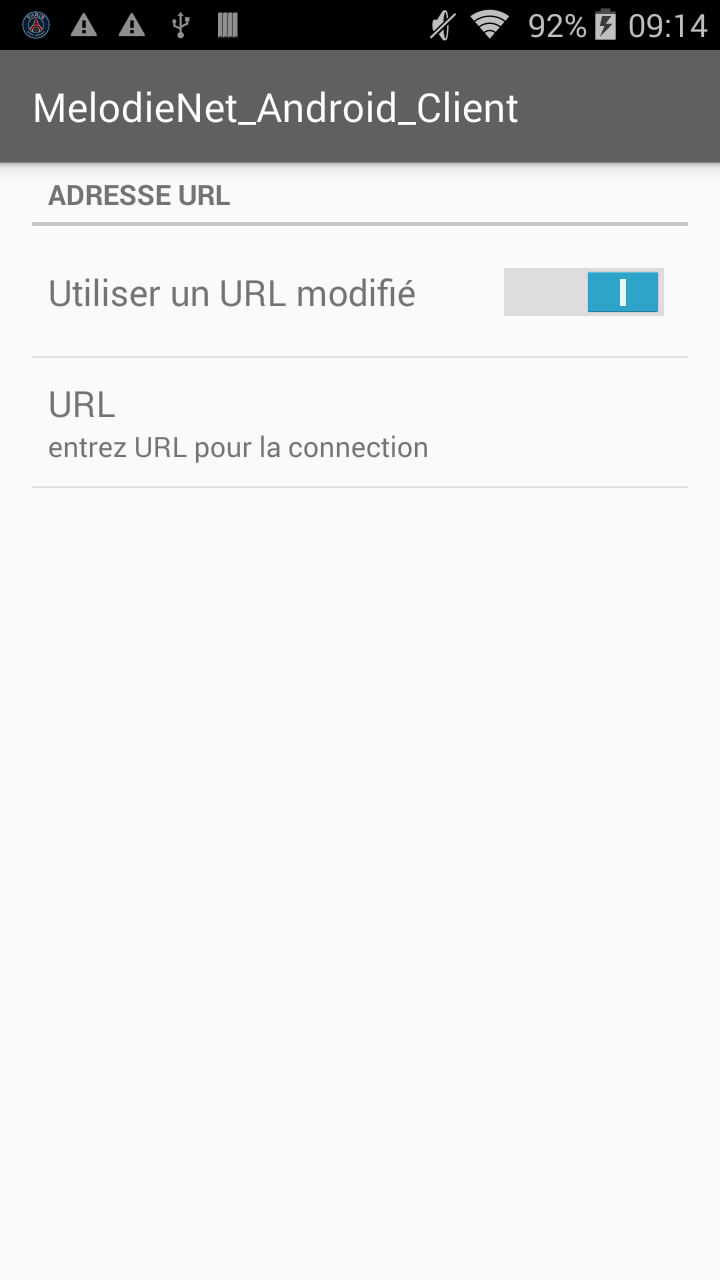
 

Figure 13 : Page de connexion Figure 14 : Vue des paramètres

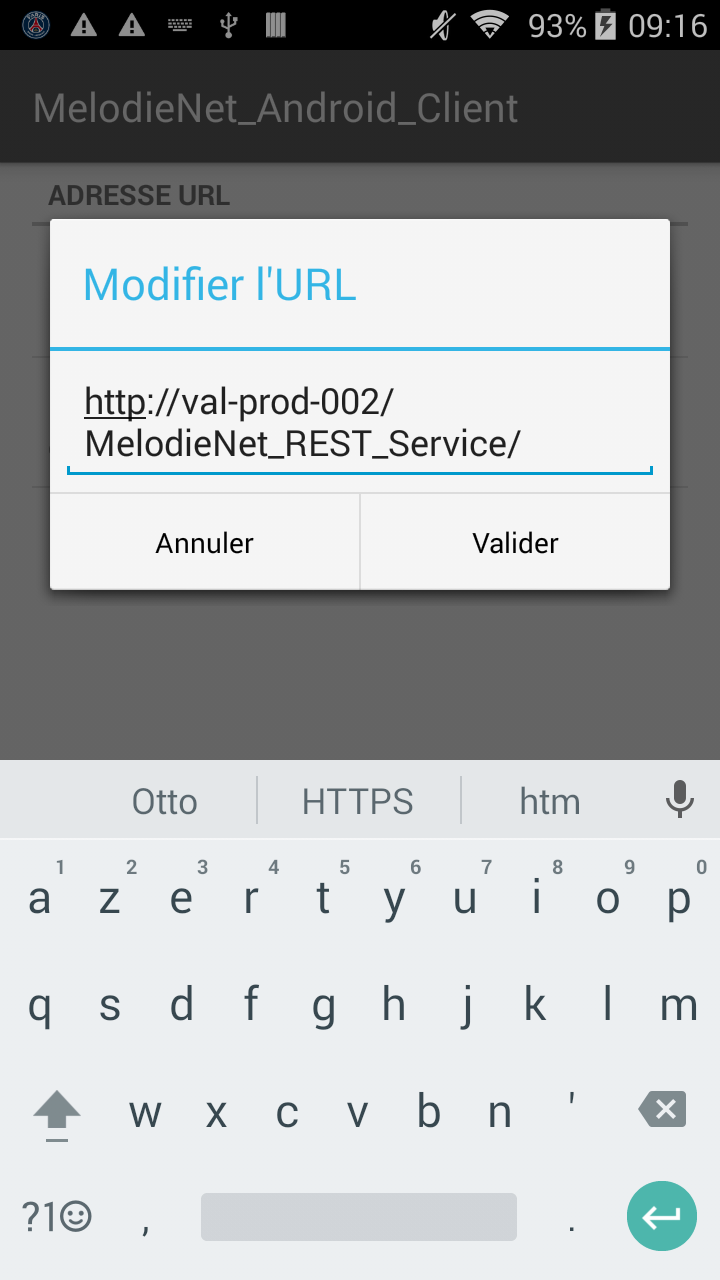
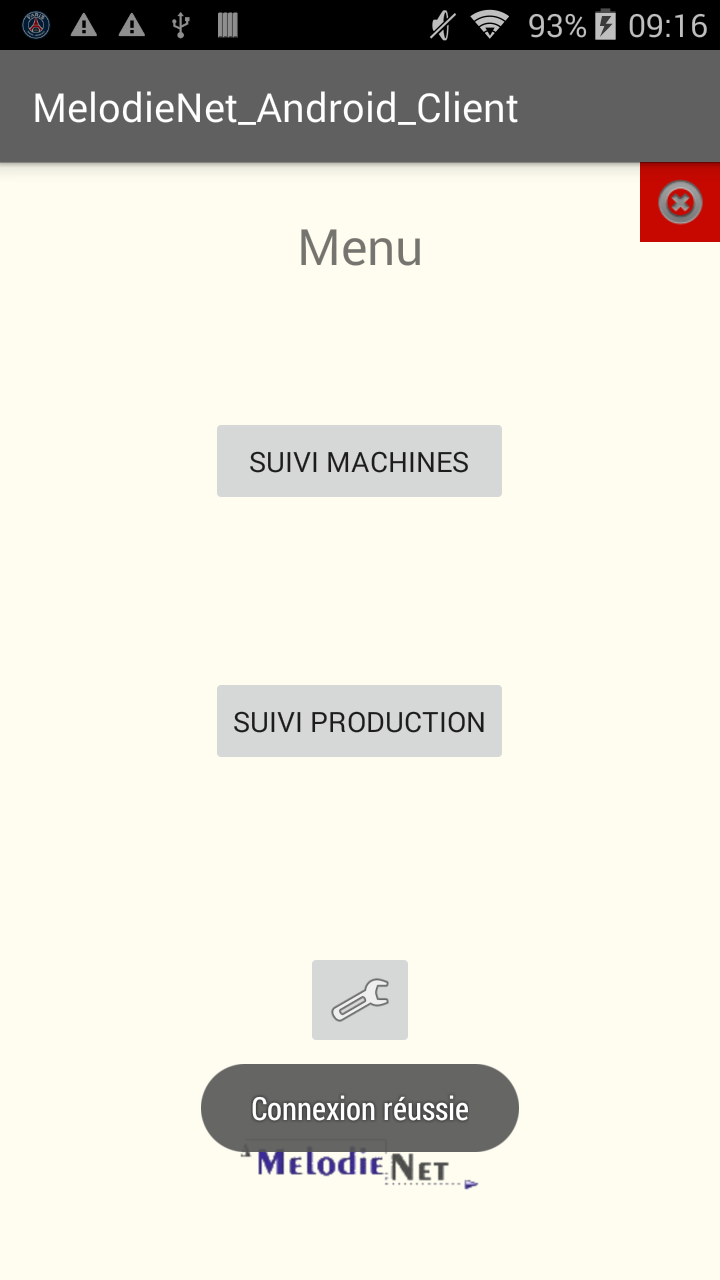
 

Figure 14 : modification de l'URL Figure 15 : menu de l’application

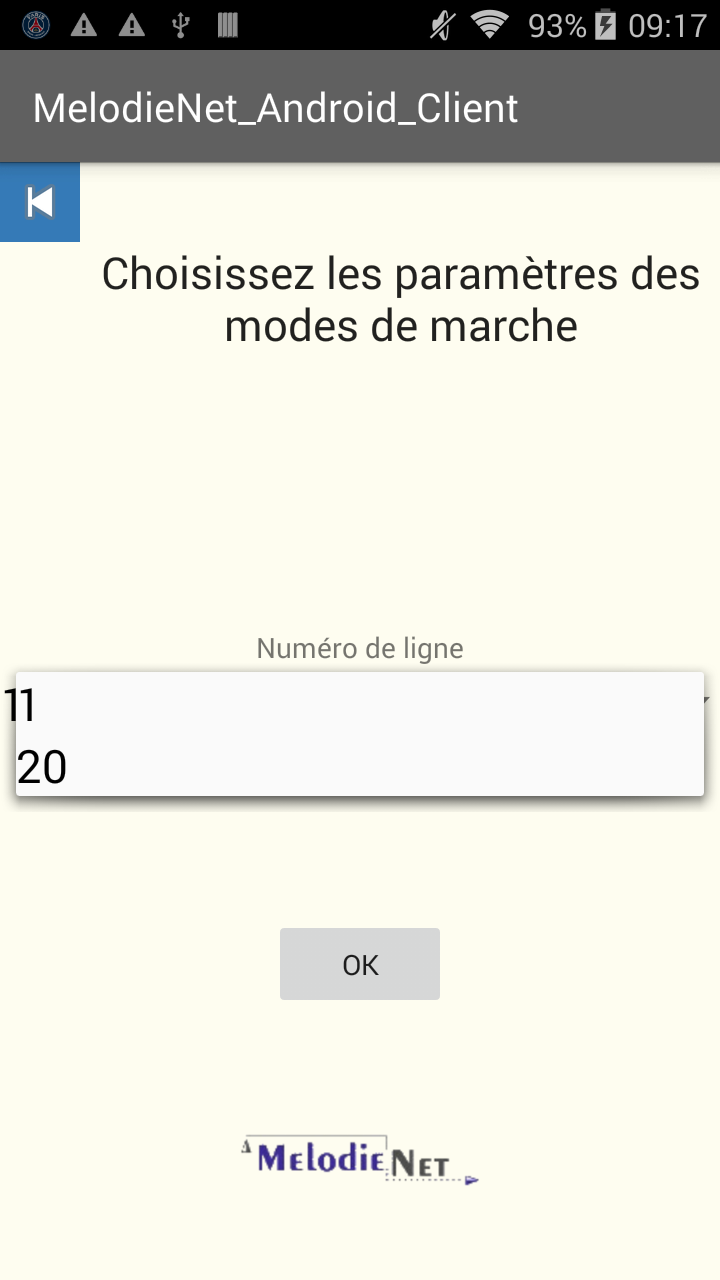
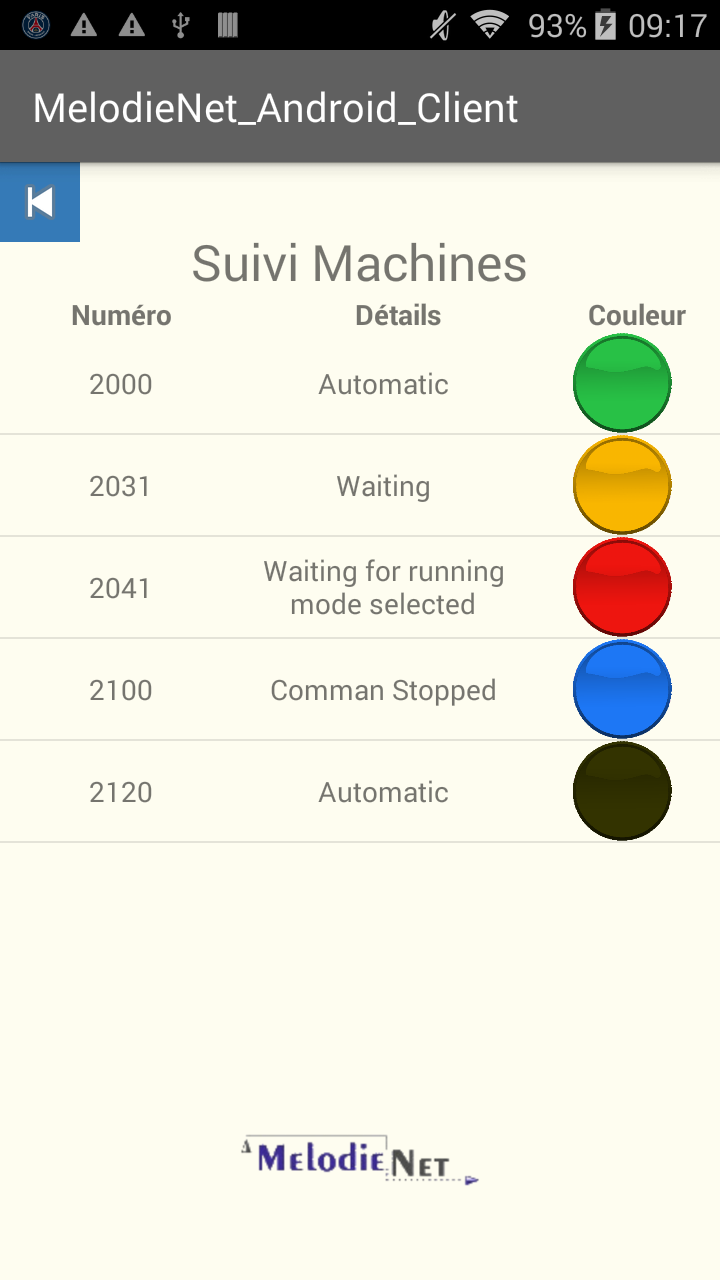
 

Figure 16 : paramètres des modes de marche Figure 17 : Suivi machine

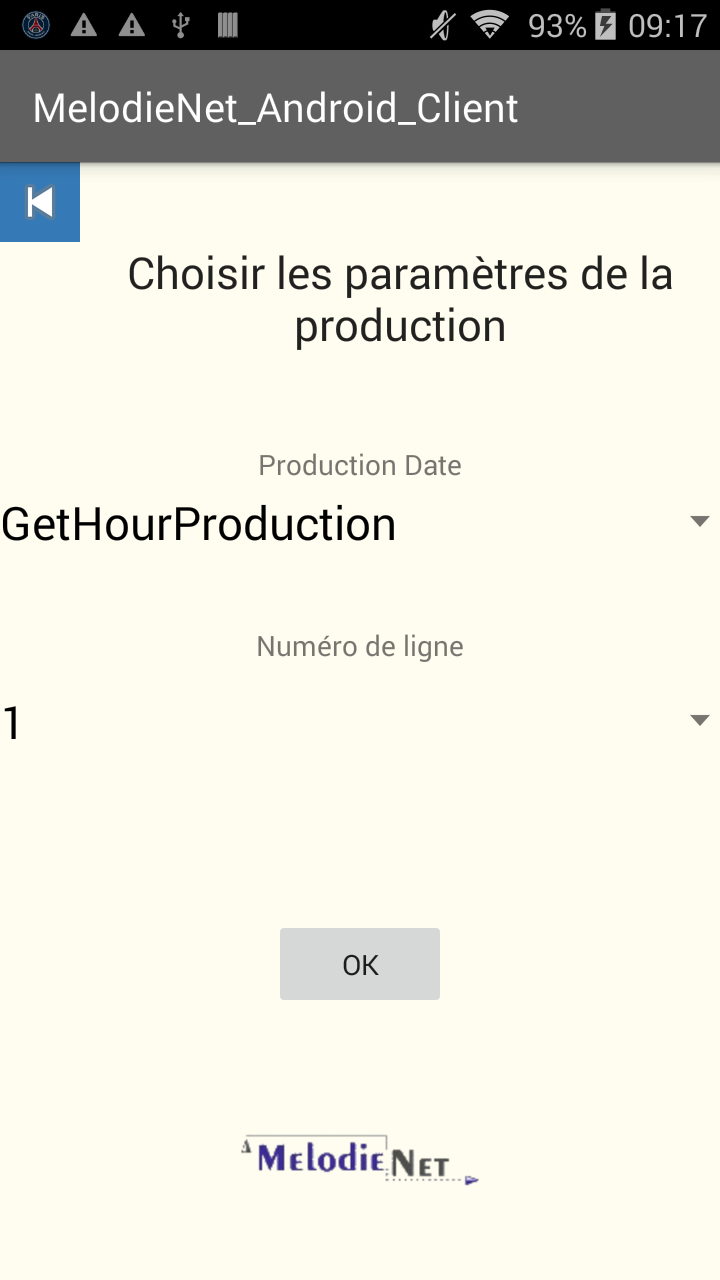
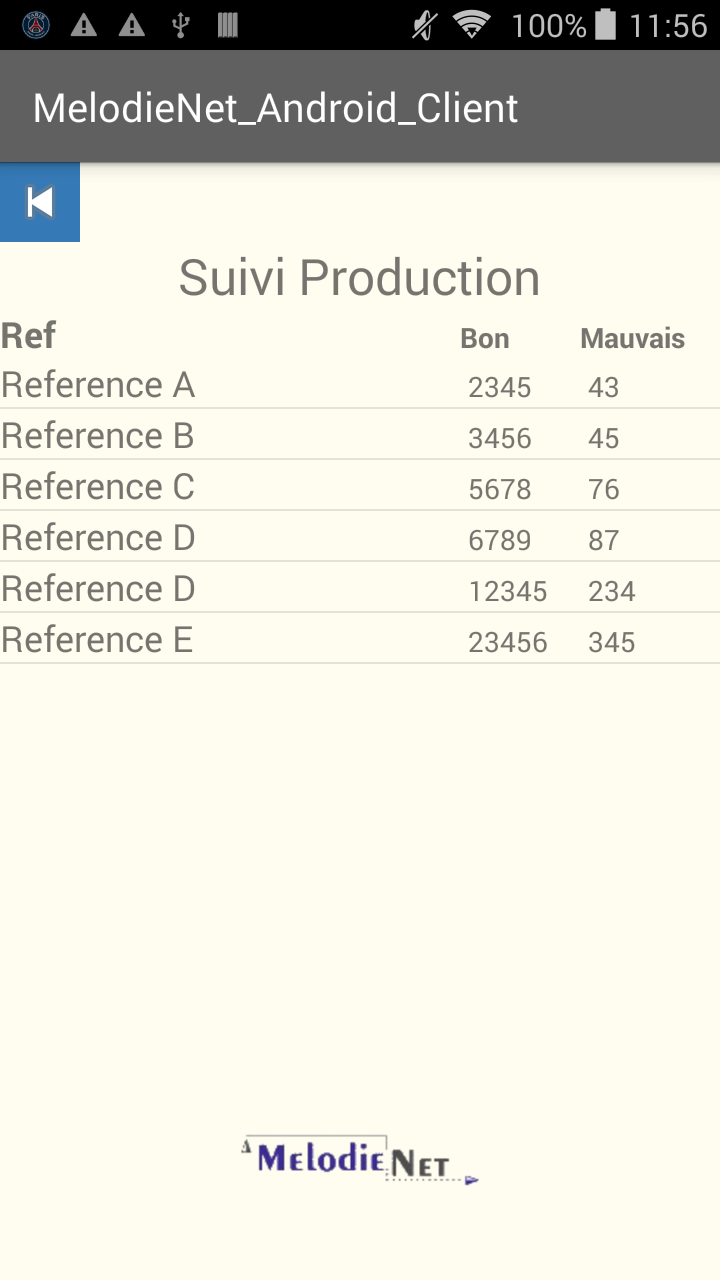
 

Figure 18 : paramètres de la vue production Figure 19 : Suivi de la production