RAPPORT DE STAGE

Découverte d'une Entreprise par du Développement, des Tests et Expérimentations sur un Système Embarqué pour Automobile

RAPPORT DE STAGE

Découverte d'une Entreprise par du Développement, des Tests et Expérimentation sur un Système Embarqué pour Automobile

Lieu d'accueil

WindRiver SRL Galați
str. Al. I. Cuza, nr. 41
800001, Galați
Romania

Tuteur en entreprise

Mme DIMITRIU Luminita
Directrice
à WindRiver SRL Systems Galati

Tuteur de stage

Mme Cathie-Anne SCHNEIDER
Enseignante
à L'IUT Robert Schuman

UNIVERSITÉ DE STRASBOURG

IUT Robert Schuman

WIND™ AN INTEL COMPANY

Remerciements

Je tiens à remercier :

- **Mme Cathie-Anne Schneider**, pour m'avoir permis d'effectuer ce stage exceptionnel dans un pays étranger.

- **Mme Luminitsa Dimitriuv**, pour m'avoir permis d'intégrer WindRiver et nous avoir apporté son soutien dans toutes mes démarches.

- **Alexandru Coja** et **Nico Feican**, qui m'ont fait découvrir tout ce qu'il y avait à découvrir de Galati et de la Roumanie, pour m'avoir aidé plus d'une fois avec des conseils sur la Roumanie et rendu cette expérience à l'étranger exceptionnelle grâce à leurs attitude avenante et accueillante.

- **Mme Emilia Pecheanu,** qui a fait tout son possible pour rendre notre séjour agréable et sans encombre du début à la fin jusque dans les situations les plus délicates, qui a toujours été disponible quand j'avais besoin d'elle et toujours avec le sourire.

Sommaire

Table des matières

Nb : Les mots explicités dans le glossaire sont soulignés et marqués d'un astérisque dans le dossier.

Introduction

Depuis la fin du XX^{ème} siècle, l'informatique n'a cessé de se miniaturiser et de s'autonomiser pour créer des systèmes embarqués de plus en plus performants. C'est dans ce marché que WindRiver prospère en continuant à développer des technologies toujours plus novatrices. La succursale dans laquelle j'ai eu la chance de travailler durant 10 semaines était un parfait exemple d'entreprise dynamique moderne, à la philosophie moderniste et en tête du développement informatique d'aujourd'hui.

L'objectif que l'on a attribué à mon stage a donc été d'en **d'acquérir un maximum de connaissances sur les rouages et les mécanismes d'une telle structure et l'informatique en général.** Par ailleurs, la philosophie de travail des employés, nourrissant un développement exponentiel de l'entreprise, a permis d'élargir les pistes de réflexion. Dans ce contexte, les décalages culturels, m'ont également permis de m'enrichir et d'explorer cet environnement, si différent de la France, dans lequel j'ai eu la chance d'évoluer. Nous pouvons alors formuler l'objectif de mon stage en ces quelques questions :

« Comment fonctionne une entreprise de dimension internationale et comment y vivent ses employés ? Comment se déroule un projet au sein de cette entreprise ? Que retirer d'une telle expérience à l'étranger ? »

C'est donc à travers de petits projets qui m'ont étés donnés, que j'ai répondu aux besoins de l'équipe que j'ai intégré.

J'ai ainsi pu avoir une vision globale du développement d'un projet informatique en plus de la hiérarchie à la base de l'entreprise[1]. J'ai également pu acquérir des compétences dans une technologie nouvelle durant le développement d'un outil tout comme j'ai analysé le du cycle de vie d'une telle application dans un projet conséquent. J'ai finalement j'ai aussi pu observer et expérimenter l'utilité et l'importance des cycles de tests sur une application.

Il faut préciser que j'ai eu la chance d'exercer mon stage en Roumanie. J'ai donc non seulement pu assimiler de l'expérience professionnelle mais également une expérience personnelle très enrichissante au travers de mes interactions avec mes collègues hors de l'entreprise, mon camarade exerçant le même stage que moi et simplement la population autochtone.

[1] Certaines informations comme les noms des employés ou le nom des Clients/ Projets seront volontairement substitué, afin de respecter la NDA à laquelle je me suis engagé.

La mission dans son environnement professionnel

WindRiver dans sa globalité

On peut immédiatement attester de l'étendue de l'expertise de WindRiver rien qu'avec leur site internet :

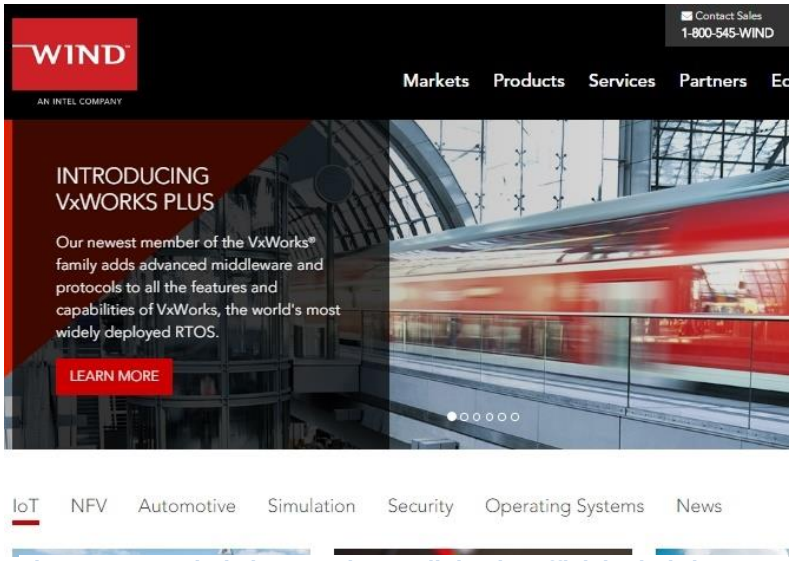


Figure 1 : Extrait de la page d'accueil du site officiel WindRiver

En effet, les 6 premières catégories affichées sur leur page d'accueil sont les **6 domaines d'expertise principaux** de WindRiver.

« IoT » est ici pour « Internet of Things*» et NFV pour « Network Function Virtualization*». Certains de ces domaines font partie de la demande Informatique de ces dernières décennies[1],[2].

Historiquement, WindRiver était une compagnie indépendante, bien développée dans ces derniers domaines et qui avait la force d'être responsable d'**un tiers du marché global** des systèmes embarqués* [3] et système d'exploitation en temps réel. Un élément intéressant, à mon sens, pour exprimer le point auquel WindRiver est compétent et **à la pointe** dans ses domaines d'expertises est **WindRiver Helix** :

WindRiver Helix est la philosophie de WindRiver quant au développement de l'IoT. C'est-à-dire le fait que chaque étape du cycle de la communication inter-appareil par Internet ainsi que la collecte et l'exploitation des données échangées soit un domaine d'intérêt et un point à exploiter, à développer. *[Voir annexe 1 : WindRiver Helix]* Cette philosophie responsable de plusieurs choix d'orientation de développement dans la compagnie, notamment WindRiver Rocket[4], n'est que la partie « IoT » des 6 domaines d'expertise sur lesquels se concentre WindRiver.

Il se trouve que la société Intel, basé principalement sur du hardware*, pouvait profiter de posséder une entreprise experte en software* et la filiale acquérait donc WindRiver en 2009 pour la somme de 884 millions de dollars.

WindRiver n'a pas pour autant perdu de son autonomie ou du adapter ses objectifs, Intel est juste devenu une sorte de client privilégié à qui développer et concevoir des systèmes embarqués, ceci afin de garder un contrôle sur toute la conception de leur production.

WindRiver sépare ses différents domaines d'expertise de manière géographique. Par exemple « Security » qui comprends la Sécurité et Défense Militaire ne sont développés qu'aux USA, la Simulation de Systèmes en France et L'Automobile en Roumanie. Cela n'empêche pas les autres zones géographiques d'exécuter des tâches différentes de leurs pseudo-attributions, nous le verrons plus loin.

J'ai eu, pour ma part, la chance d'effectuer mon stage en Roumanie, le WindRiver dont je vais parler est donc spécialisé dans la conception de systèmes embarqués à usage automobile. Il est intéressant de noter que « WindRiver System SRL Galati » est **situé dans le seul RDC** (Centre de Développement Régional) informatique de Roumanie. Ceci a pour conséquence de faire de la localisation de WindRiver Galati, le **lieu de toutes les conceptions et développements informatiques d'Europe**. C'est-à-dire que dès qu'un système d'exploitation, un driver*, une fonctionnalité un peu lourde devra être conçue et développée, elle sera **conçue et développée en Roumanie**.

WindRiver Galati est divisée en trois bâtiments, WR1 et WR3 sont situé dans la même rue à 100 mètres l'un de l'autre et WR2 se trouve 3 rues plus loin. WR1 est principalement dédiée à l'assistance et le support technique des clients WindRiver ainsi que de l'administratif pur tandis que WR2 est axé sur la Simulation Informatique. WR3 ne contient que le développement complet, la conception et la réalisation de projets de systèmes embarqués pour automobile. C'est dans ce dernier que j'ai effectué mon stage.

Une philosophie de travail

Différents type de management d'employés sont trouvable en Roumanie, un des managements possible est de garder quelque employés clef qui vont former les nouveaux à leurs postes. Mais de leur imposer des conditions de travails impossibles et éreintantes qui feront démissionner ces derniers avant qu'ils ne montent en grade ou ne se fasse virer par l'entreprise. Un autre management possible est de chérir ses employés afin de les rendre fidèles à l'entreprise, motivé et efficace sans les épuiser. Tout ceci pour minimiser le turn-over et stimuler la créativité et l'enthousiasme de ses employés.

Figure 2 : Nourriture offerte par l'entreprise

C'est dans cette seconde optique que WindRiver a investi ses ressources. Le bien-être de ses employés est ainsi assuré en fournissant des viennoiseries, différents types de cafés et de thés, différents types de fruits, différentes sortes de snacks et boissons à volonté le tout gratuitement. Les locaux bénéficient également d'une attention particulière, ils sont spacieux et impeccables grâce à la présence constante des techniciennes de surface, une table de Ping-Pong avec matériel fournis ainsi qu'un babyfoot et un jeu de fléchette y sont présents, une large salle à manger avec lave-

vaisselle, ustensiles et vaisselles fournis. *[Voir annexe 2 : Locaux de l'entreprise]* WR3 a la chance, contrairement aux autres bâtiments de WindRiver Galati, d'avoir une terrasse avec panorama appréciable et un grand jardin aménagé pour les pauses, le tout animé par une chienne et un chat qui participent à garder le moral de l'équipe. Tous les bâtiments WR à Galati ont les mêmes avantages, le jardin en moins. Ces locaux ne sont pas utilisés à mauvais escient car tous les mois, une fête est organisé en l'honneur de tous les employés promus, nouvellement mariés, les naissances ou évidement les anniversaires du mois. Il s'organise même des festins improvisés quand un évènement spécial se présente durant un projet (j'ai par exemple pu assister à un festin de plusieurs cartons de pâtisseries et plateau de plats de viande parce qu'un bug bloquant et critique du projet avait été réglé). Bien évidemment, tous les employés à qui j'ai parlé mon affirmé que WindRiver était la meilleure entreprise dans laquelle travailler à Galati (Galati étant classée dans les 10 plus grandes villes de Roumanie) voire de toute la Roumanie.

Pour continuer dans cette optique agréable et sans stress, WindRiver a décidé d'opter pour des **horaires libres**. Cela s'applique en prenant en compte que les employés travaillent pour accomplir leur tâche de la journée, il ne tient qu'à eux de répartir leur charge de travail comme ils le souhaitent avec les pauses dont ils ont besoin. Le bon fonctionnement d'une équipe et la bonne participation d'un employé est assuré de manière non contraignante puisqu'elle repose sur la motivation et la volonté de ses employés au sein d'une équipe et, au pire, de l'autorité du chef d'équipe.

Figure 3 : Boisson et nourriture offerte

Ceci est peut-être la partie la plus intéressante de cette philosophie de travail, là où la plus part des horaires flexibles serait de types 4 jours de 10h par semaines ou autre spécifications réclamant un nombre fixes d'heures[5], WindRiver laisse une **réelle liberté pour ses salariés.** Certains employés peuvent arriver aux alentours de 10h30 et repartir à 18h30 simplement parce qu'ils se sentent mieux dans ce rythme de vie spécifique. Cette liberté de management de temps possède de nombreux bénéfices[6] et je n'ai jamais entendu parler d'un quelconque point négatif à cela durant toute la durée de mon stage.

Une culture d'entreprise américanisée

Ce mode de fonctionnement peut paraitre idyllique, exceptionnel ou irréel, cependant beaucoup d'entreprises internationales avec une très bonne implantation dans leurs domaines font de même. WindRiver Galati bénéficie donc de cette culture d'entreprise américanisé qui est d'entretenir la motivation et la **loyauté des employés**. En particulier dans le secteur Informatique qui comprends plus de passionnés du métier et possède un **lien très fort** entre **la productivité** des employés et **leur contentement** de leur environnement de travail. On retrouve beaucoup cette intégration de l'entreprise dans

l'intimité ou le quotidien des employés de façon très américaine quand on observe les locaux et le matériel : Des posters de motivation un peu partout, des T-shirt « WindRiver Rocket » ou « WindRiver Pulsar Linux », des tasses WindRiver.

Figure 4 : Exemple de posters de motivation

Les différentes équipes organisent une réunion une à deux fois par semaine et souvent des lieux incongrus comme un couloir, la salle à manger, l'open-space d'une autre équipe. Ceci à cause du manque de salle de réunion qui sont continuellement occupées. Cela peut sembler paradoxal étant donné que les horaires sont libres, que les équipes continuent de rester synchronisées et à avancer ensemble sans se bloquer, et pourtant il ne se passe pas un jour sans que la salle de réunion soit occupée ou que l'on assiste à une mise au point entre collègue dans la cour de l'entreprise ou entre deux open-space.

Une hiérarchie complète et internationale

[Voir schéma page suivante]

Nous pouvons trouver à la base de la hiérarchie de WindRiver, des projets et des équipes de programmeurs. Une équipe peut avoir plusieurs projets mais cela n'arrive que rarement et un projet concerne souvent plusieurs équipes. Ces équipes sont réparties sur les projets par domaines qui nécessitent une attention spéciale à tel étape du projet (une équipe pour La sécurité, une équipe pour le hardware etc…). Elles comprennent toute un Chef d'équipe, appelé **Team Lead**, qui veille à ce que tous les membres de l'équipe aient du travail et le fasse correctement.

Ces équipes doivent rendre compte à une **Direction Technique (Technical Lead - TL**) et un **Project Manager (PM**). La TL n'est pas toujours présent dans le développement d'un projet, cependant elle présente un avantage dans sa productivité car c'est elle qui aura le **dernier mot** sur les décisions de type technique et évite ainsi de longs débats contre productifs. Un PM quant à lui, est responsable de différentes équipes sur un ou plusieurs projets, il gère les ressources humaines d'un projet et organise la répartition globale des équipes sur les domaines du projet dont il faut s'occuper. Il aussi responsable de la communication avec le client. Il est sous la direction d'un directeur géographique qui ne fais que de l'administratif et d'un responsable situé en France (pour le cas de WindRiver Galati), lui-même sous les ordres d'un responsable de WindRiver USA. Puis l'on trouve le Vice-Président et finalement le Président de WindRiver.

Mon bâtiment comprend 6 PM et 2 projets importants. J'ai été intégré dans la « Testing Team » du projet « Hinata », elle comprend une dizaine de personnes et a pour Team Leader **A.** , le PM de cette équipe est **B.** .

La plupart de la jeunesse de WindRiver est composé d'étudiants qui alternent projets de cours, les cours en eux-mêmes et travail. La plupart des employés sont à WindRiver grâce à un stage qui s'est bien déroulé et, en général, y font carrière. Ils obtiennent ainsi régulière de la main d'œuvre fraiche et prête à former. Sachant que c'est une des plus importante entreprise de Galati, on comprend pourquoi leur culture d'entreprise est si fructueuse.

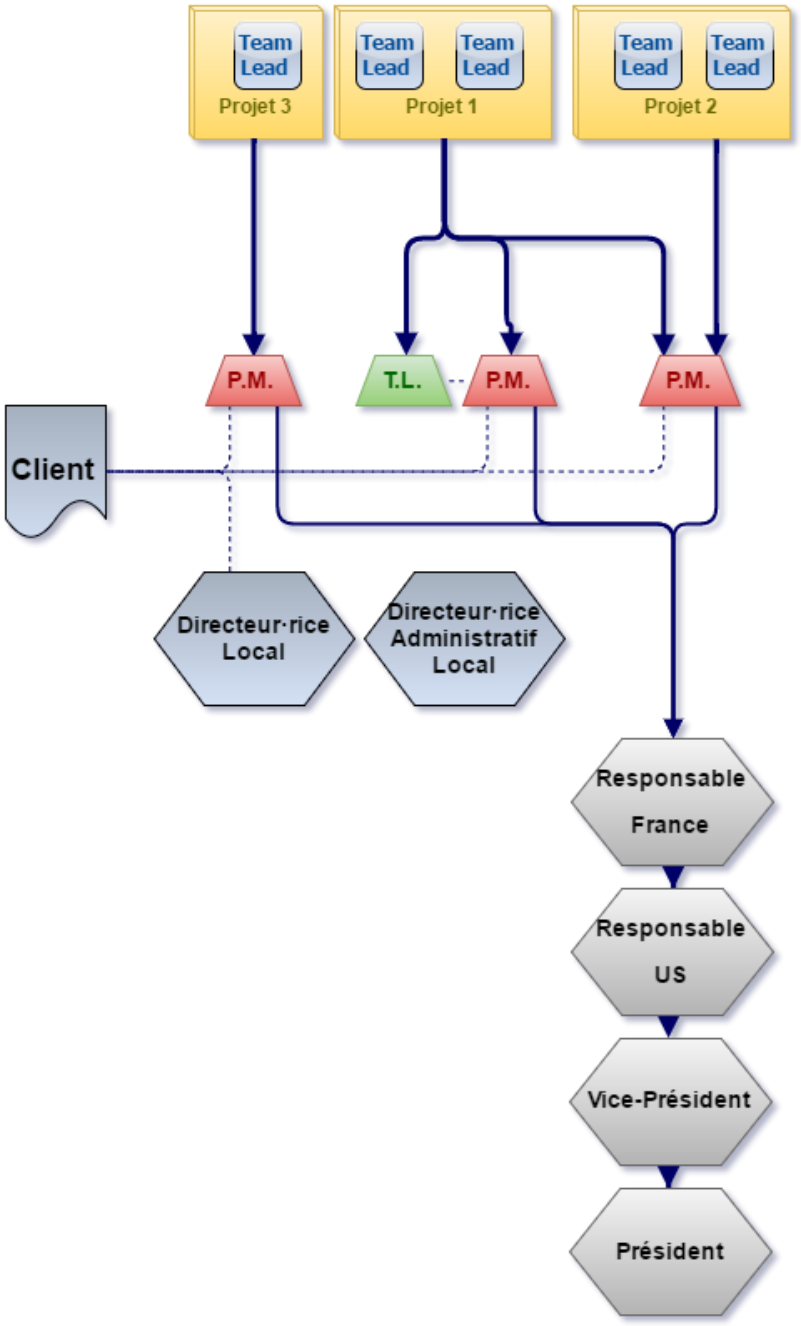
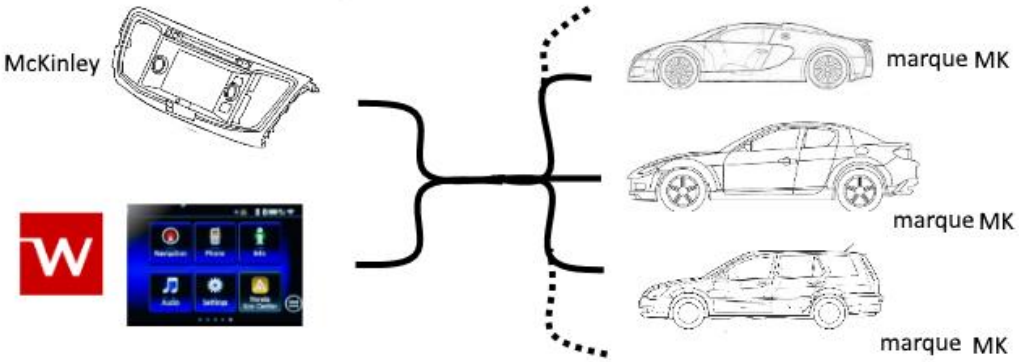


Figure 5 : Schéma d'exemple de la hiérarchie globale de WindRiver SRL Galati

Les projets de WindRiver3

Le premier projet est le projet « **McKinley »** et consiste à développer le système embarqué complet d'un tableau de bord dont le hardware a été créé et assemblé par Honda. Le tableau de bords ne sera cependant pas destiné à être intégré dans un seul modèle de véhicule mais la majorité d'entre eux tant que l'interface développée restera satisfaisante et compatible.

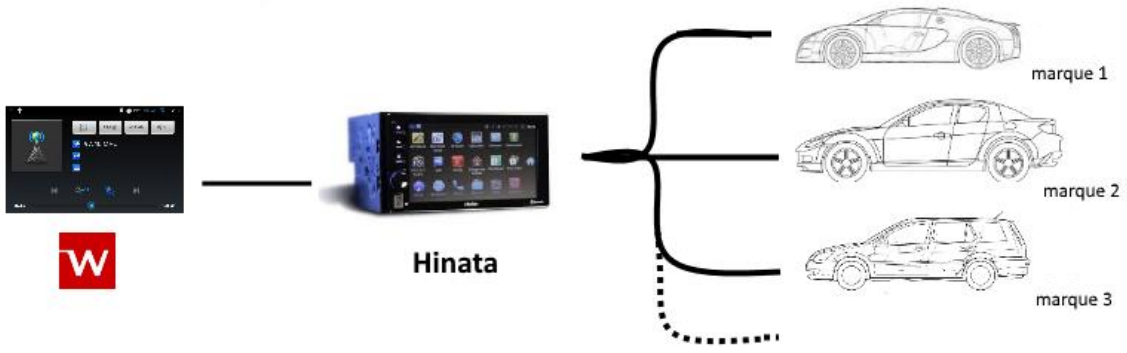
Figure 6 : Schéma d'exploitation du projet McKinley



Ce projet mobilise la majorité des ressources de WR3 car il concerne ¾ du bâtiment de 2 étage, soit **93 personnes** (il y a en tout 130 personnes sur ce projet), il mobilise également **6 PM** qui prennent en charge en tout **17 équipes**. Le stade du projet quand je suis arrivé était plutôt avancé mais manifestement en retards car, selon des témoignages recueillis par des participant du projet, les PM ont dû demander une rallonge sur la deadline* quelques temps après mon arrivés.

Le deuxième projet développé dans ce bâtiment est le Projet **« Hinata ».** Il est moins ambitieux et ne comporte qu'une vingtaine de personne, c'est aussi un système embarqué cependant non pas pour une marque de voiture mais pour une marque de tableau de bords. Ici le modèle de voiture est indéterminé, le système embarqué n'est fait que pour un modèle de tableau de bords et ce tableau de bords sera acheté par d'autres entreprises qui l'intègreront dans leurs modèles.

Figure 7 : Schéma de l'exploitation du projet Hinata



Ce projet n'est pas un nouvel OS à partir de zéro comme McKinley mais la mise à jour du précédent OS déjà opérationnel et exploité. Les équipes de ce projet sont chargées de faire une nouvelle version « plus rapide, plus légère, plus complète ». En plus de ce projet, certaines équipes de ses équipes travaillent à la maintenance de l'ancienne version jusqu'au moment de la mise à jour.

Le déroulement du stage

Temps d'adaptation et découverte

Comme je l'ai expliqué précédemment, il n'y avait pas vraiment de besoins existants auquel devait répondre mon arrivé. En attendant d'avoir une mission à proprement parler, j'ai passé quelques jours à m'informer sur le fonctionnement de l'équipe dans laquelle j'allais m'intégrer et à lire de la documentation à propos de l'environnement de développement.

Fonctionnement de l'équipe

En globalité

Quand je suis arrivé la première fois, B. m'a juste demandé si je préférais développer ou tester. Après une petite discussion il m'a annoncé que puisque que son but était que j'acquiers le maximum d'expérience sur l'entreprise et l'informatique en général, j'allais intégrer la « Testing Team » d'A. pour simplement réaliser des petits projets en fonction des besoins de mon chef d'équipe.

Dans un premier temps, A. , mon chef d'équipe, m'a parlé du fonctionnement de l'équipe et leur rôle dans tout le projet. Son équipe fait partie des trois équipes en charge du projet Hinata : La **"BSP Team",** la **"Testing Team"** et la **"Android Application Team"**. On peut s'attendre à ce que ces noms soit assez explicites et coïncident avec l'ensemble des tâches que les équipes vont exercer... Mais ce ne sont en fait quasiment que des pseudonymes qui englobent bien plus de travail qu'ils en ont l'air.

La "BSP Team" se charge de tout ce qui est **code de bas-niveau** et drivers, BSP signifie "Board Support Package". Le mot Board* désigne la partie **hardware**, donc les composants électroniques physiques du tableau de bord :

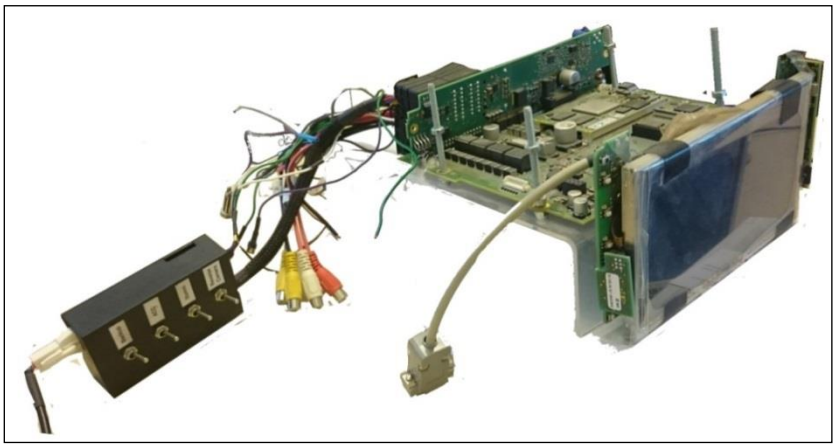


Figure 8 : Tableau de bord à nu

Support Package désigne tous le code permettant aux couches supérieures* du projet de communiquer avec le board. L'équipe BSP s'occupe donc la **bonne communication entre le board et l'OS**, pour que chaque application puisse exploiter pleinement les capacités techniques du board. Elle s'exécute en rédigeant le code qui va servir d'interface entre des applications simples (couche de haut niveau d'abstraction de code), la machine virtuelle* de l'environnement de cette application, et le board précédemment cité. Le code se doit d'être sans faille car il sera **la fondation** de tout l'OS qui va se construire dessus. C'est pour cela que ce domaine possède sa propre équipe dédié.

La "Android Application Team" est celle qui s'occupe de toute la couche apparente du projet, elle développe toute les applications du système ainsi que les réglages possibles, tout ce qui figure dans le cahier des charges du client. La version que j'ai pu tester possédait une quinzaine d'applications fonctionnelles à ce moment du projet. Cette équipe s'occupe également de toute la partie **UI** (User Interface) de l'application, le design des applications et leur ergonomie est décidée principalement par le client.

La "Testing Team" quant à elle, prend en charge de **tester** chaque nouvelle fonctionnalité afin qu'elle convienne aux attentes du client, que son code soit viable pour le projet et respectueux des conventions et règles établies depuis le début du projet. Cette pratique est appelée de la "**review de code"**. L'équipe va parcourir du code avec certains outils spécialisés (Je sais qu'ils utilisent SonarQube sur le projet McKinley) et identifier s'il y a des optimisations possible du code tel que plus de commentaire à certains endroits, des fonctions superficielles ou au contraire trop grandes qu'il faut morceler. Le "reviewer" de code tente de voir si un l'autre programmeur qui devra travailler avec le dit code aura la tâche facile ou non. De plus, l'équipe s'occupe de tout ce qui concerne l'**intégration** des nouvelles fonctionnalités dans le projet principal, en faisant des test donc, et en créant des demandes de patch à travers une plateforme spécialisé en cas de non-conformité ou simplement de bug. Puis évidement elle s'occupe de l'intégration de ces patchs dans le projet.

Mon équipe d'accueil

J'ai rejoint les sept personnes qui composaient la Testing Team dès mon arrivé à WindRiver. **Aucun membre n'avait de spécialité** particulière dans le projet mis à part E. qui s'occupait uniquement de développer la stratégie de test et la mettre en place. J'ai d'ailleurs eu quelques contacts avec lui au moment où je devais programmer les tests de ma mission principale. Le reste des membres de l'équipe recevaient leur tâche en fonction des besoins actuels du projet, ces tâches pouvaient concerner n'importe quels domaines de l'Informatique.

La technologie au cœur de la mission

Ces trois équipes sont donc responsables du développement intégral de la nouvelle version du système embarqué pour ce tableau de bords spécifique. Seulement ils ne partent pas de zéro. Pour comprendre sur quoi cette équipe se base pour développer son système embarqué, Il faut savoir qu'un système embarqué comprend plusieurs parties (schéma) :

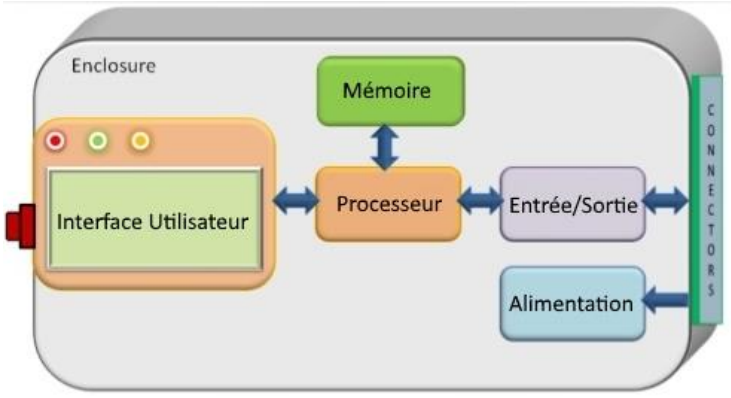


Figure 9 : Schéma de composition d'un système embarqué [7]

Dans la partie virtuelle de ce système embarqué Il y a à la toute base l'OS. L'OS est une couche qui s'occupe de gérer les ressources disponibles (CPU*, Mémoire*) et les rendre accessibles aux applications/processus actives/fs qui s'occupe de l'Interface Utilisateur (UI). Or cette partie est extrêmement complexe et il peut nécessiter **des années de développement** pour concevoir rien qu'un prototype. C'est pour cela que l'équipe se base sur un OS spécialement conçu pour les systèmes embarqués : **Android**.

Android

Durant ma période d'adaptation, mon chef m'a donné quelques ressources afin de me préparer pour une mission future. J'ai donc commencé à parcourir le site d'Android[8] pour comprendre globalement son fonctionnement.

Android est une des nombreuses distributions de Linux. Elle se base sur un système à plusieurs utilisateurs, chacun ayant des droits particuliers. Chaque application installé se voit attribuer un UserID. Ce qui fait la particularité d'Android, c'est que ses applications sont écrites en langage **Java**, c'est-à-dire qu'une application lancé bénéficiera d'une **machine**

virtuelle* qui **facilitera la communication** entre l'application et l'OS. Cet environnement personnel permet aussi à l'application de s'isoler des autres pour une indépendance et une sécurité complète.

Donc pour préciser les activités des équipe citées précédemment : La BSP Team s'occupe donc de rédiger le code en **C** qui permet à l'OS de communiquer avec les composants qui ne seraient **pas déjà reconnu par la base Android**, ils **complètent** cette base. L'Android App. Team rédige donc toutes les applications demandées par le client grâce aux outils fournis par Android et la **Testing Team** s'occupe de tout **rassembler sous le même build*.** La majorité des communications semi-professionnelle se font par Skype et Webex pour ce qui est purement professionnel. Les informations se passent aussi lors de nombreuses réunions ad-hoc.

La syntaxe des patchs et du code

A. m'a aussi donné de la documentation sur comment écrire du code Java[9] et en général la syntaxe à respecter quand on collabore avec une communauté[10], cela m'a été très bénéfique car je voulais savoir s'il y avait un manuel ou une norme à laquelle se soumettait tous les employés de WindRiver quand à la manière de rédiger du code homogène. Il semble donc que ces deux sources soit les seules lignes directrices que chacun essaye d'appliquer.
Quatre points m'ont particulièrement marqués car je n'y avais jamais prêté attention avant.

- **Utiliser les usages des commentaires Javadoc Standard :**

Car jusqu'à maintenant je faisais que commenter avec des « // » classique à chaque fois que je voulais me rappeler de quelque chose, à la manière d'un post-it. Seulement, on peut lire ici qu'il est bien meilleur de commenter **chaque méthode* non-triviale** avec un « /** […] */» en expliquant ce que fait la fonction de façon explicite avec à au moins une phrase au minimum avec un verbe à la troisième personne.

- **Ne pas attraper d'exceptions* génériques**

J'ai pourtant toujours eu l'habitude de faire de la sorte, on ne nous a jamais vraiment imposé, même dans les cours qui en parlaient, d'attraper les bonnes exceptions pour faciliter l'implémentation du code ou sa complétion par une personne tierce. Cette habitude peut pourtant être fatale dans la conception d'un code en entreprise, dans un projet d'ampleur moyenne.

- **Indenter avec 4 et 8 espaces**

Cette déclaration de bonne pratique est commune à la fois aux règles de programmation Kernel* et Java, ce qui la rends étrangement importante. Étrangement car je ne connais personne de mon entourage qui ne code pas en utilisant des tabulations. Mais il semblerait qu'il est de très bon usage d'indenter avec des espaces.

- **Encapsuler plus systématiquement**

Que ce soit pour une simple boucle while* triviale ou une encapsulation try catch*. Le fait tout encapsuler dans des petites fonctions permet de réduire grandement la taille d'une fonction principale et, grâce au nommage de l'encapsulation, d'augmenter significativement la lisibilité du code. C'est ce genre de notions que l'on ne peut assimiler que dans le milieu professionnel.

Ce ne sont que quelques points facilement assimilable parmi la myriade d'éléments d'amélioration de la lisibilité de son code, mais les respecter demande déjà un certain effort et une certaine rigueur qui n'en sera que **bénéfique dans mon avenir professionnel et mes collaborations futures.**

Une première mission concrète

Présentation de la mission

Ma première mission s'est révélée avoir une certaine utilité car le client demandait à avoir un **outil pour configurer précisément le son de l'appareil**. C'est ainsi que j'ai eu ma première consigne de manière moyennement conventionnelle :

"We need an android application that will change an xml file when the user changes some slides and then presses on the "Save" button.
The app will have 3 tabs, 1 for each: Main Volume, Alert Volume and BT_Voice Volume.
Each tab will have the same content but will change a different part of the xml file.

Tab contents:
- *1 slider for volume, having 33 steps, from 0-32, step=1, default value=0.*
- *1 slider for FixedBoost having 3 steps(24.08, 48.16, and 60.2), default value is 48.16.*
- *2 sliders for MainVolMax one for the integer part and one for the fractional part of it's value*
  - *The integer part will have one slider that will have 25 steps, from 0 to 24, default value is 0 and the step is 1.*
  - *The fractional part will have one slider that will have 100 steps from 0 to 99, default value is 0 and the step is 1.*
- *1 slider for MaxLoudBoost having 20 steps from 0 to 19, the default is 0 and the step is 1.*
- *2 slider for VoldB one for the integer part and one for the fractional part of it's value*
  - *The integer one having 100 + MainVolMax(125 max) steps, from -100 to MainVolMax value, step is 1.*
  - *The fractional one will have 100 steps from 0 to 99, default value is 0 and the step is 1."*

Le mail comprenait le document XML qui devait servir à stocker les valeurs des sliders* *[Voir annexe 3 : Extrait du document XML des valeurs de volumes].* Le format XML indique que tout le document sera écrit grâce à des balises de nom arbitraire imbriqué les unes dans les autres :

```
<balise1>
    <balise2></balise2>
</balise1>
```

Et à la fin de l'encastrement de balise on pourra trouver une valeur :

```
<balise1>
    <balise2> valeur </balise2>
</balise1>
```

On peut également rentrer des valeurs dans des champs de valeurs dans l'ouverture d'une balise, ceci permet des créer des regroupements de valeurs facilement accessibles et de manière totalement libre :

```
<balise1 numero_de_balise='9'>
    <balise2> valeur </balise2>
</balise1>
```

Pour permettre une meilleure compréhension du rapport plus tard, je me dois de préciser ce que j'ai compris de ladite consigne au moment de sa réception.

On peut constater dans le document qu'une seule ligne comprend les champs de valeurs que j'étais censé modifier (VoldB, FixedBoost, MainVolMax, MaxLoudBoost), j'en ai donc déduis que les autres enchainements de valeurs n'étaient utiles que pour un autre service qui utiliserait ce document (ce qui, après coups, ne me parait pas cohérent avec ce que j'ai lu sur le fonctionnement d'une application Android). De plus, le premier slider volume **n'a aucune correspondance dans ce champ de valeurs**, j'ai pensé que peut-être que A. prévoyait par la suite de faire interagir ce slider avec le système, donc je n'ai pas posé de questions. Finalement j'ai vu que sur les trois balises d'identifiant « main », « alert » et « bt_voice », **seulement « main » avait cette suite de champs de valeurs**. J'ai donc modifié le premier « Step » des deux autres balises pour avoir aussi un champ de valeur à modifier.

Mon but a donc été de faire cette suite de **7 sliders qui modifieraient la première ligne de la balise main,alert ou bt_voice voulue du document xml**. J'avais dans l'idée de passer d'un onglet (Tab) à une autre en exerçant un glissement du doigt.

Pour assurer le bon déroulement de mon développement et me venir en aide si j'étais trop en difficulté ainsi qu'avoir une trace de ma progression ; je devais rendre à A. **un petit rapport** sur mon avancement dans l'application toutes les fin de journées. Cette requête était à la fois un **service pour lui** et un **avantage pour moi** car mes mini-rapports comprenaient trois parties :

- DONE
- TODO
- TOFIX

Ce qui me permettait à moi de savoir où j'en étais en revenant le jour suivant, même si je pensais l'être à la fin d'une journée avec la tête remplis de code. Ce petit rapport me **permettait** de **clarifier mon organisation et ma progression** pour moi-même, et ainsi **d'être plus efficace** le lendemain. C'est une habitude qu'avait essayé de nous inculquer notre professeur de système et réseaux en nous faisant rédiger notre rapport petite section par petite section, mais cet exercice en stage m'a vraiment persuadé de l'efficacité de ce genre de pratique et je l'applique déjà plus au quotidien.

La préparation pour la mission

Comme l'objectif de mon stage était d'apprendre de manière « **relax** », pour citer A. , Il m'a conseillé de me familiariser avec l'environnement en commençant par la plus basique des réalisations quand on découvre une nouvelle technologie : **afficher un texte « Hello World** ». Cela requérait déjà de :

- Savoir sur quel IDE* j'allais développer

- Savoir quel est le premier élément à créer qui s'affichera tout seul

- Savoir comment afficher mon application sur le board que l'on m'a fourni.

A. m'a informé que l'équipe, et beaucoup d'autres employés, ont commencé à développer sous Android avant que Google ne développe un IDE spécialisé. Ils avaient donc pour habitude de développer sous Eclipse avec un plug-in pour Android. Cependant quand j'ai cherché quel IDE pouvait me convenir, Android Studio était sorti depuis assez longtemps pour être stable et surtout développé par IntelliJ, la même entreprise qui a créé certains de mes IDEs préférés. Mon outil de travail a donc été **Android Studio**.

En ouvrant l'IDE la première chose se présentant à moi était quel type d'application je voulais faire. J'avais, au final, assez **peu de spécifications** sur le type application exact que A. et ses clients me demandais, je ne savais pas encore si cela allait être une simple application à lancer depuis le menu principal ou si elle allait être incorporé dans une application plus importante (choses qui se sont précisé bien plus tard). J'ai donc sélectionné dans la liste des modèles que me proposait l'IDE ce qui se rapprochait le plus de l'image de l'application que j'avais en tête, en pensant que je la modifierais à chaque étape passé.

Comment est construite une application Android ?

Le type de modèle proposé par Android était nommé **« Tabbed Activity** », je l'ai donc validé et ai observé tous les fichiers nécessaires à la création d'une telle application s'écrire automatiquement. Puis, avant de programmer ce petit « Hello World », j'ai passé beaucoup de temps à **comprendre** ce que Android Studio venais de générer par défaut et comment ces **différents éléments interagissaient** entre eux. C'est une étape assez importante au démarrage du développement sous un nouvel environnement et j'ai découvert beaucoup de mécaniques de fonctionnement d'une application Android.

La meilleure façon d'expliquer comment fonctionne une application est avec un simple exemple. Il faut savoir qu'une application est au format **.apk** (Android PacKage). C'est donc un **paquet** de différents types de ressource.

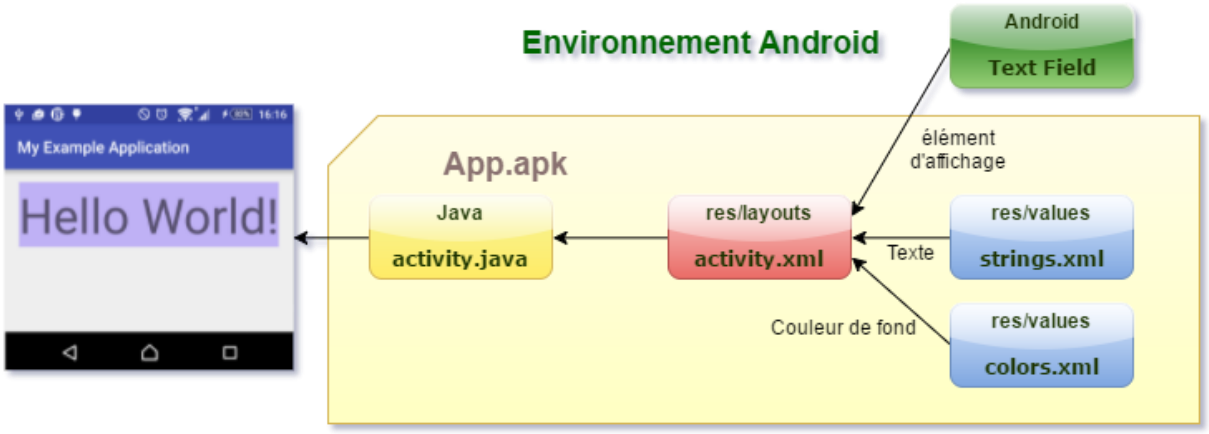


Figure 10 : Schéma d'organisation des ressources dans une application Android simple

Contrairement aux différentes architectures d'application que j'ai pu utiliser, tous les types de ressources utilisés pour composer une application sont explosés et triés entre eux.

Dans notre paquet, il y a une partie **Java** qui contient **tout le code**. Il décrit le fonctionnement du programme ainsi que la façon dont les différent **layouts** (et fragments) seront imbriqués ensembles. Les **layouts** sont des **fichiers .xml** qui **contienne**, balise par balise, **l'ordre d'affichage**, l'encastrement ainsi que **les propriétés d'affichage des différents éléments** de notre application. Les propriétés écrites peuvent avoir un caractère de design (hauteur, largeur, couleur, etc....) ou pratique (ID de l'élément, version de l'élément à utiliser, etc…). Ces **éléments sont fournis par l'Environnement Android**, donc en dehors de notre .apk. Dans les propriétés d'affichage de ces éléments, on trouve aussi les **valeurs** que doivent prendre nos éléments, que ce soit des **valeurs** de couleurs, de taille, de texte, etc… Les valeurs définies par le développeur sont regroupées ensembles dans des fichier .xml de même type de valeurs. Nous aurons donc par exemple un fichier strings.xml contenant toutes les chaines de caractères qui apparaîtront dans nos applications. Et au moment de placer dans notre **layout** un **élément zone de texte**, nous lui préciserons l'ID du texte qu'il doit contenir, ce texte étant défini dans le fichier **strings.xml**. Nous lui préciserons aussi la couleur de son fond défini dans **colors.xml**. Ainsi, si l'on change juste le texte du fichier strings.xml, le texte changera dans l'application mais nous n'aurons rien touché de l'élément d'affichage en lui-même.

Ce procédé permet de pouvoir **aisément apporter des modifications** comme les tailles des caractères, toutes la palette graphique d'une application ou simplement passer tout le texte d'une application d'une langue à une autre. **La modularité de l'application est ainsi considérablement augmentée**. De plus, un élément important d'une application (comme la partie appareil photo, l'affichage d'une carte ou simplement l'affichage principal de l'application) est appelé **« Activité »**. Une application peut posséder plusieurs activités (chacune avec son code source) et **cette activité peut s'échanger** d'une application à une autre, c'est-à-dire que si l'application rendre disponible son activité appareil photo, une autre application pourra ne lancer que cette activité au sein de son propre affichage et faire comme si l'appareil photo faisait partie de son fonctionnement de base. **Cette modularité et cette interchangeabilité sont de grandes qualités, surtout dans le domaine de la**

compatibilité inter-appareils. C'est que qui me motive à **continuer de me développer dans ce langage** et surtout à **en tirer un maximum d'expérience** tant que j'en ai l'occasion car il y aura encore certainement beaucoup de débouchées avec cette technologie dans le futur.

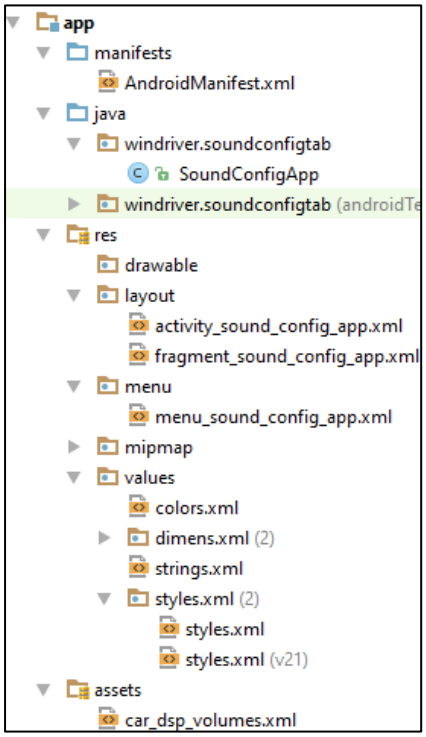


Figure 11 : Image de l'organisation des ressources de l'application par Android Studio

En plus de ces éléments basiques, une application Android comprend également un fichier appelé **AndroidManifest.xml**. Ce fichier sert à indiquer au compilateur* quelle version d'Android utiliser pour l'assemblage, quelle version de Java, quel type de build utiliser (Gradle ou Maven). Le type de build est en fait très important car c'est lui qui va fournir une grande partie de ses ressources et ses fonctionnalités comme **l'Environnement Android** que le développeur va utiliser dans son application. À une certaine époque, il était courant d'utiliser Maven et beaucoup de projets de WindRiver SRL Galati fonctionnent encore avec ce dernier. Cependant à un certain point de l'existence d'Android Studio, Gradle a été nommé build par défaut de l'IDE et s'est donc complètement intégré dans son fonctionnement[11]. Maven est considéré comme obsolète à présent, ceci se remarque à la grande différence de fonctionnalité disponible entre les deux builds[12]. Ceci à évidement poser quelques problème au moment de rendre mon application quelque fois mais nous nous en sommes très bien sortit grâce à l'expérience d'A..

Tout fichier auxiliaire à l'application, comme notre fichier xml, sera mis dans un dossier **« Assets »**. Ce qui nous donne l'organisation affichée sur la figure 11.

La connexion avec le board

Une fois cette partie éclaircie, un nouveau questionnement s'offrait à moi. *Comment allais-je afficher sur le board le petit Hello World que j'avais codé ?* Le problème n'a pas mis beaucoup de temps à se résoudre, merci à A. et l'intuitivité d'Android Studio. Mais il m'a cependant **fait découvrir un aspect très intéressant d'Android**, ce qui est une bonne étant donné l'objectif de mon stage qui est, rappelons-le, **d'apprendre le plus possible des technologies que j'utilise.**

Afin de faciliter le développement d'applications Android, Google a intégré à Android Studio divers émulateurs* d'appareils Android. Ceci permet à n'importe quel utilisateur/développeur lambda pour pouvoir tester son application sans avoir d'appareil physique disponible. Avoir l'appareil ne suffit, il faut un moyen de communiquer avec. Et il se trouve que Google a eu la bonne idée de développer un outil de communication par ligne de commande qui laisse une **très grande liberté de contrôle à l'utilisateur**. Cet outil s'appelle **ABD (Android Debug Bridge)**, il se base sur une communication client*->serveur*->daemon*. Le client est le programme adb.exe que l'on lance dans notre invite de commande , le serveur, lancé par Android Studio, localise tous les appareils disponibles pour

une connexion adb et établit toutes les connections et protocoles nécessaire à la communication entre le client et l'appareil. **Le daemon** est lancé dans l'appareil (virtuel ou physique) et se charge **d'exécuter toutes les commandes reçu dans l'appareil**. La liberté d'action que le développeur dispose viens du fait que ce daemon peut simuler des actions sur l'appareil comme appuyer sur n'importe quelle touche (accueil, retour , Lettre, Numéro, etc…) ou exécuter les commandes Linux classiques ou spécifique à Android, comme installer une application par exemple.

```
Target device: sony-c6903-BH9072VX06

Installing APK: C:\WindRiver\SoundConfTab\SoundConfigTab\app\build\outputs\apk\app-debug.apk

Uploading file to: /data/local/tmp/windriver.soundconfigtab

Installing windriver.soundconfigtab

DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/windriver.soundconfigtab"

        pkg: /data/local/tmp/windriver.soundconfigtab

Success

Launching application: windriver.soundconfigtab/windriver.soundconfigtab.SoundConfigApp.

DEVICE SHELL COMMAND: am start  -n "windriver.soundconfigtab/windriver.soundconfigtab.SoundConfigApp" -
a android.intent.action.MAIN -c android.intent.category.LAUNCHER

Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=windriver.soundconfigtab/.SoundConfigApp }
```

Figure 12 : Extrait du retour de la console au moment de lancer une application depuis Android Studio

C'est exactement comme ça que fait Android Studio quand le développeur lance l'application qu'il vient de développer. Il établit une connexion adb avec l'appareil choisit (`target device`), transfert le .apk compilé (`Installing APK`), exécute la commande d'installation (`pm install`) et finalement exécute la commande d'exécution (`am start`) comme si un utilisateur l'aurait fait en appuyant sur l'icône.

L'affichage par fragment

À peine ais-je commencé à vouloir écrire mon code que je me suis trouvé confronté à une nouvelle problématique : J'avais appris comment fonctionnait une application de base, simple mais pas comment fonctionnait une application avec **plusieurs pages** accessibles par glissement de doigt. En regardant dans le code que m'avait généré Android Studio, j'ai remarqué que l'IDE avait déjà utilisé une astuce plutôt complexe, aux premiers abords, pour afficher ses 3 pages.

Le Dossier **Layout** comportait 2 layouts : *activity_sound_config_app.xml* et *fragment_sound_config_app.xml,* le premier layout comprenait dans ses balises un élément **ViewPager**. Le second ne contenait qu'un texte d'exemple avec une chaine de caractères contenant un élément variable (dans le fichier string.xml, une valeur *section_name* qui avait pour valeur "%1$s").

Au lieu d'avoir un fragment pour chaque page, Android avait préparé dans le *activity[…].xml* (qui sera l'affichage principal) un endroit (balise ViewPager) ou insérer des fragments de layout selon la section désirée. Pour ce faire et en ceci économisant de la puissance de calcul, le code java utilise une classe* **Fragment** et une classe **FragmentPagerAdapter**.

La manière dont ces deux classes sont exploitée est intéressante car c'est une application directe d'un cours de DUT. Nous n'utilisons pas les classes précédentes directement mais en créons deux nouvelles ayant les propriétés* désirées héritant* de ces dernières. Nous nous retrouvons donc avec deux classes :

- Une classe **PlaceHolderFragment** (Fragment à Espace Réservé) héritant de Fragment. Elle prépare un Objet* à partir du Layout *fragment[…].xml* en remplaçant les espaces réservés (dans notre cas la chaine de caractères "%1$s") par leurs valeurs correspondantes. Grâce à cette nouvelle classe nous avons un **Fragment** incorporable à volonté dans notre application et qui possède le code requis pour compléter notre *fragment[…].xml*.
- Une classe **SectionPageAdapter** héritant de FragmentPagerAdaptater va servir à stocker les pages créés et ne pas avoir à les recréer une nouvelle fois au cours du cycle de vie de l'application. Ici Android les nomme **Sections** car dans le contexte de notre application ce sont réellement des sections (Section main, Section Alert, Section Bt_Voice). En temps normal, un fragment serait utilisé au sein d'un layout comme une petit partie de ce layout (donc comme un fragment) mais vu qu'ici on les considère comme **une page entière**, le nom de Section est plus approprié et évite la confusion si l'on compte par la suite utiliser d'autres Fragments.

Je n'avais pas compris ce principe au début du développement mais je m'en suis approché et ai pu continuer dans ma progression. C'est bien après avoir fini l'application finale que j'ai compris le fonctionnement de cet affichage par fragment. Ceci a été pour moi une preuve du progrès que j'avais fait durant mon stage.

Développement de l'application

Une fois les principes de base de développement d'application J'ai donc commencé à développer ma première version de l'application telle que je l'imaginais.

Comme à chaque fois que je développe une application avec un fonctionnement nouveau pour moi, j'ai essayé de mettre en place un bon découpage de fonction dans l'intention de pouvoir revenir sur des fonctions/ fonctionnalités qui devrait être changé par la suite, mais j'ai aussi pensé à rendre facilement modulable l'organisation du layout de mes sections.

L'interface utilisateur

J'ai pris en compte l'astuce d'Android et ai décidé de n'en faire qu'un qui sera afficher sur les trois pages, en modifiant la classe PlaceHolderFragment pour avoir les noms des sections dans le champ titre au lieu d'un numéro de section. L'avantage était que si l'on "swipait" d'une section à l'autre durant un changement non-enregistré dans les sliders, on gardait ces changements. Pour faciliter la réutilisabilité du layout, surtout compte tenu du nombre d'éléments qui allait être affiché, j'ai pris soin de correctement encapsuler chaque slider.

Un Slider tel que je l'imagine dans le contexte de mon application est composé de 4 à 6 éléments selon le cas (décimale ou non) :

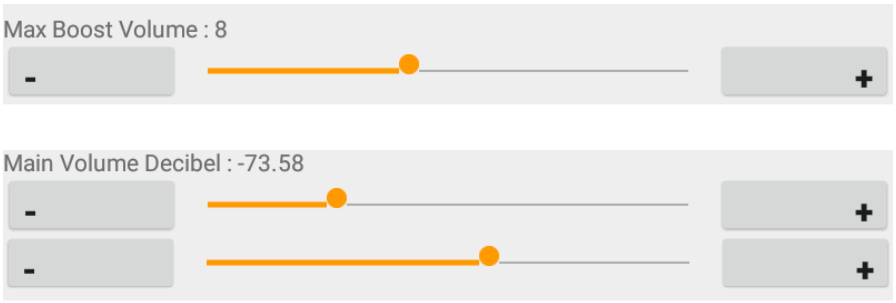


Figure 13 : Capture d'écran des deux types de groupement de sliders possibles

Le slider en lui-même, les deux boutons de précision, et le texte de valeur. Si l'on veut un partie décimale il faut alors avoir un deuxième assemblage de sliders et bouton, mais qu'un seul texte de valeur.

Pour la clarté et la réutilisabilité de l'application j'ai décidé de créer un groupe qui contiendra uniquement le sliders et ses boutons dans un **groupe linéaire "slider"**. Ensuite un groupe avec le texte avec la valeur et un ou deux groupes linéaires.

Puisqu'on pouvait distinguer 3 types de sliders : les maximums, les volumes simples et les volumes de décibels, j'ai trié ces types de sliders en groupe qui contenaient tous un titre.

Le xml étant une hiérarchie simple de balise imbriquée, il est facile de représenter l'organisation que je viens de décrire sous la forme d'un schéma, pour une meilleure compréhension.

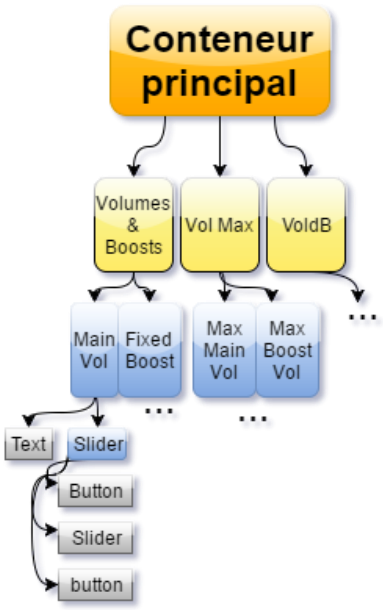


Figure 14 : Schéma de l'organisation visuel de l'application

Après quelque heure de manipulation et de prise en main, je suis finalement arrivé à un résultat satisfaisant. . *[Voir annexe 4 : Capture d'écran de l'application une fois le design visuel terminé]*

Algorithmes et principes de fonctionnement de l'application

La première version de mon application était assez simple et j'ai commencé par établir un algorithme naturel de l'exécution de mon application :

- Charger les valeurs du XML
- Afficher ces valeurs
- Mettre en place des actions des Sliders et Boutons :
    - Bouton de précision appuyé => Incrémenter/décrémenter le slider associé
    - Slider change de valeur => Changer le texte associé
    - Slider MaxVolDb change de valeur => Changer le maximum du slider VolDb
    - Bouton de sauvegarde appuyé => Écrire les valeurs des sliders dans le fichier xml.

On se doute que la partie "attribution des valeurs" et "enregistrement des valeurs " va nous demander d'accéder aux éléments contenant les valeurs (les sliders). Pour permettre aux développeurs de sélectionner des éléments dans un layout au plein de l'excecution d'une application, La bibliothèque Android fournit une fonction que j'utiliserais tout au long de mon stage et certainement dans le reste de la programmation Android que je ferais dans ma vie professionnelle : `FindViewById(idAChercher)`

Cette fonction nous permet à partir d'un ID, que l'on a précisé au moment de la rédaction du layout, de **récupérer un Objet** que l'on pourra utiliser, dans le cas d'un slider on pourra en extraire sa valeur courante ou lui en attribuer une. Il faut prendre en compte que cette fonction est une fonction de recherche et elle va par conséquent parcourir une liste d'ID avant de trouver le bon. Une fonction de parcours et de recherche **consomme systématiquement une quantité non-négligeable de ressources.** Je désirais faire les choses bien et comme que je travaillais sur un système embarqué, je m'étais mis dans l'idée de d'économiser le plus de puissance de calcul possible même si ce n'était pas requis. Pour ce faire, il ne fallait pas que le programme exécute cette fonction de manière intempestive et non contrôlé. Ceci pourrait mener à une perte significative de performance si celle-ci se retrouvait exécutée en boucle au seins d'une fonction quelconque. J'ai donc décidé de concevoir mon programme de manière à ce que cette fonction lourde ne soit utilisée qu'au lancement du programme et n'exécute toutes les opérations lourdes **qu'une fois et une seule**.

Le travail de préparation que j'avais fait en prévision de changement s'est donc avéré utile car j'ai ainsi pu ainsi économiser de la puissance de calcul en rassemblant ces fonctions dans un dictionnaire de slider facilement accessible par référence sans opération lourde. Ceci facilitait la relecture de mon code et ne faisait exécuter la fonction de recherche qu'une fois au lancement de mon programme. Cette fonction se nomme `loadCorrespondences()`.

Les première modifications et corrections

Une fois la première version testée (à la main) et livrée à mon chef j'ai pu avoir ma **première expérience de révision de code** dans le développement d'une application. Il y avait quelques aspects de l'application qui ne lui convenait pas (du moins au client avec qui il échangeait) et quelques requêtes qui mise ne pratiques ne fonctionnaient pas. Toutes ces petites modifications à apporter ne sont pas venues tout de suite, mais au fur et à mesure de la conception de la deuxième version de l'application. À partir de ce rendu, nous sommes rentrés dans un cycle de patch / application / résultat qui m'as vraiment appris sur la façon de corriger son application à la demande d'un client.

L'outil de révision de code et le cycle de vie d'un patch

Il existe une multitude de technologies, de principes, de philosophie afin d'organiser le travail commun dans le cycle de vie d'un projet, certains élément se retrouve assez souvent comme le tableau de tâches avec ses 5 colonnes BACKLOGS, TODO, Work In Progress,TESTING, DONE. Ce tableau peut paraitre anodin mais il est représentatif d'une équipe synchronisée avec une bonne synergie.

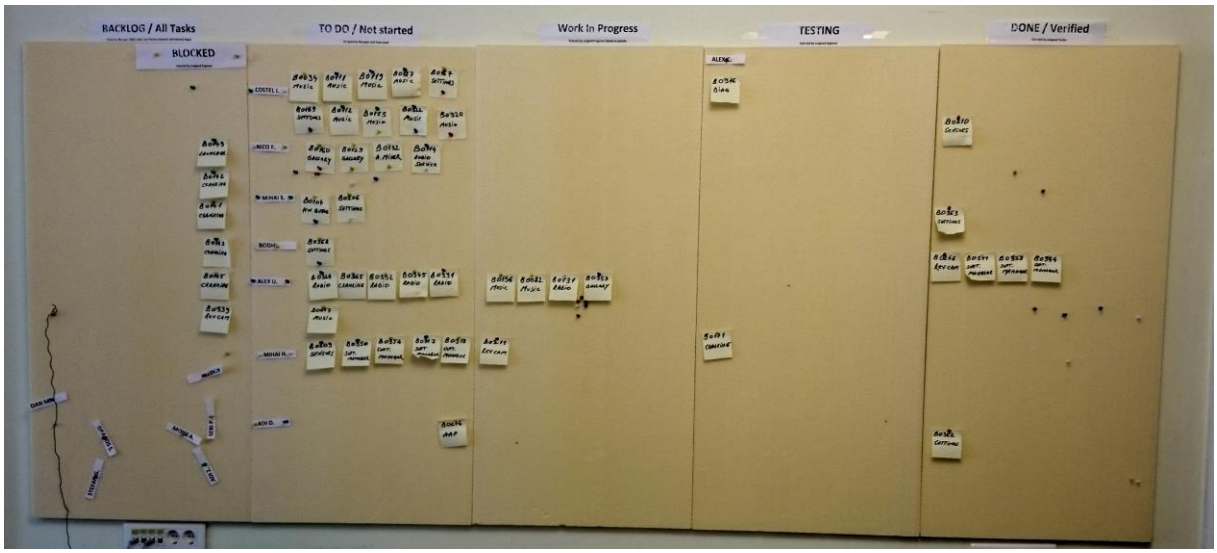


Figure 15 : Tableau de tâche de l'équipe du prjet Hinata

Je n'ai pas eu l'occasion de travailler directement avec l'outil dont je vais parler à cause de la confidentialité imposée au projet Hinata. Cela ne m'a pas gardé de me renseigner et de poser des questions à mon chef qui s'est donné un plaisir de m'informer et me faire une démonstration de cet outil.

Dans la vie d'un projet informatique, les développeurs de ce projet doivent trouver un moyen de se coordonner pour rédiger un code manière éclatée le plus efficacement possible. Nous avons pu tester certain de ces outils en DUT et l'un d'entre eux est aussi le plus connu : **Git.**

Ici, Git n'as pas sa fonction première de **système de contrôle de version,** mais pour sa fonctionnalité de générateur de **fichier de différence, dites "diff",** entre deux versions d'un code. Il existe plusieurs autres d'outils de génération de diff, mais c'est celui-là que mon chef m'a dit d'utiliser. Ce fichier .diff contiendra la description de toutes les lignes d'un code qui ont été ajouté et toutes les lignes qui ont été supprimés, dans un ordre qui préserve le travail du programmeur. Ainsi, pour avoir une version à jour d'un projet volumineux, on a juste à appliquer ce **.diff** pour mettre à jour son code plutôt que de toucher à tout ou tout réinstaller. Ici seules les lignes nécessitant une modification sont touchées et le code à jour reste intact. Cette diff, de par son caractère correctionnel, rentre dans une catégorie de code que l'on appelle **patch**.

```
diff --git a/res/layout/fragment_loudness.xml b/res/layout/fragment_loudness.xml
index 87073cd..35c244e 100644
--- a/res/layout/fragment_loudness.xml
+++ b/res/layout/fragment_loudness.xml
@@ -43,16 +43,27 @@
                         style="?android:attr/buttonStyleSmall"
                         android:layout_width="0dp"
                         android:layout_height="40dp"
+                        android:id="@+id/sbBassBoostMaxIntButtonDecr"
+                        android:gravity="start"
+                        android:layout_weight="0.1"
+                        android:text="@string/decrMax"
+                        android:textStyle="bold"
+                        android:backgroundTint="@color/colorAccentLight"
+                        android:textSize="20sp"/>
+                    <Button
+                        style="?android:attr/buttonStyleSmall"
+                        android:layout_width="0dp"
+                        android:layout_height="40dp"
                         android:id="@+id/sbBassBoostIntButtonDecr"
                         android:gravity="start"
-                        android:layout_weight="0.2"
+                        android:layout_weight="0.15"
                         android:text="@string/decr"
```

Figure 16 : Extrait du résultat d'une diff avec git. En bleu - les lignes ajoutées; En rouge - les lignes supprimées; En blanc - les lignes intouchées: En vert - les commentaires générés

Le fichier de diff en lui seul ne suffit évidemment pas à créer un cycle complet car le patch peut être défectueux en lui-même, ou les modifications qu'il apporte pourraient ruiner les modifications d'un précédent patch. C'est pour cela que WindRiver a instauré un cycle de vie spécifique au patch et surtout à un bug d'une partie du projet. Ce cycle de vie se base sur un outil essentiel à la gestion de patch et de synchronisation de code : **Gerrit.**

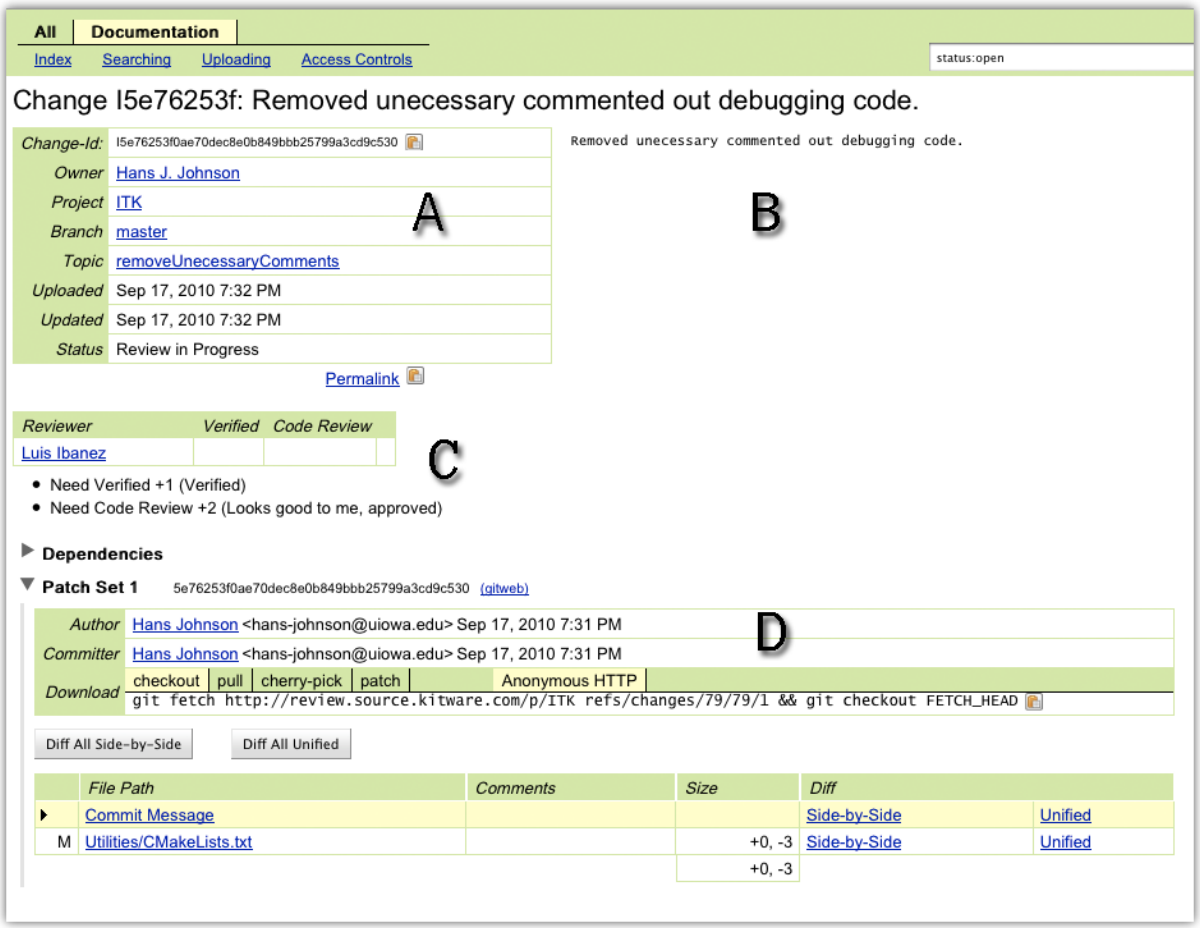


Figure 17 : Extrait de la page Gerrit d'un patch quelconque

Le bug est reporté sur Gerrit dans sa section associé. Une fois qu'il est attribué à un membre de l'équipe, ce membre va créer et proposer sur Gerrit un patch à ce bug **(A)**. Il détaillera le plus pertinemment possible l'action de ce patch **(B)** afin de pouvoir facilité la review de code. À ce moment le patch n'est absolument pas appliqué au projet mais passe par de la review de code de plusieurs utilisateurs qui vont indiquer si le patch leurs parait correct simplement du niveau code et aussi s'il corrige le bug originel **(C)**. Si tout va bien le patch est appliqué et validé sur Gerrit pour faire partit du code final. S'il ne l'est pas, les utilisateurs signalent de manière constructive et précise les lignes défaillante ou dérangeante et le patch est amélioré à sa version suivante qui recevra le même traitement **(D)**.

Ceci est la version très simplifié du cycle de vie d'un patch. Car il existe beaucoup plus de subtilité quant à la localisation des fichiers, l'impact et la gravité du bug ainsi que du patch, ce qui donne lieu à une série beaucoup plus complexe de cas et de possibilités de traitement comme on peut le voir sur ce schéma. *[Voir annexe 5 : Schéma représentatif du workflow d'un patch dans Gerrit selon Android]*

Les modifications à apporter

La nouvelle version de l'application devait changer en deux points :

- Le fait que l'application que j'avais développée utilisait des modules spécifiques à la version d'Android 6 Marshmallow (API* 23+). Et que l'application était trop "stylisée" pour ce qui était nécessaire. Il fallait que je la rende plus simple et compacte sans bouton flottants etc… Il ne fallait donc plus prendre de modèle proposé par Android comme à la première version mais partir d'une application entièrement vierge sans fioriture de glissage à droite etc…
- Le rôle du slider sans valeur xml Main Volume) cf. Présentation de la mission p.12).
- Suppression de la valeur décimale de maxVoldB

Il s'est avéré que ce Main Volume à 33 valeurs représentait les **33 valeurs que peut prendre le volume de l'appareil Android** quand on monte le son depuis ce dernier.

Figure 18 : slider de volume sur mon portable

Et les 33 lignes de valeurs pour chaque catégorie que je pensais inutiles dans le fichier xml sont en fait **les valeurs de volumes** (Main Vol) que peut prendre le son système, il manquait juste les autres champs de valeurs présente sur le premier palier. *[Voir annexe 3 : Extrait du document XML des valeurs de volumes]*

Il fallait donc que ce premier Slider Main Volume soit spécial car il désignait ces paliers dont on va personnaliser l'action sur le volume. Par exemple un palier peut avoir un son très fort alors que le palier suivant pour avoir un son très faible. Ceci selon le bon vouloir de l'utilisateur, c'est compréhensible puisque cet outil était destiné à des développeurs ou des testeurs qui voudrons vérifier précisément le son de l'appareil.

La nouvelle version et ses améliorations

J'ai donc revu le design de mon application pour mettre ce slider en avant et le rendre plus précis. Cela en utilisant des éléments plus basique et compatible (l'API la plus moderne utilisé était l'API 22, qui était compatible avec le board). La valeur de décimale de MaxVolDb a été retirée, car selon la consigne elle était censée changer le maximum de la valeur décimal de voldB. Or cette valeur décimale n'est là que pour préciser la valeur entière et la faire aller de .0 à .125 n'a aucun sens. Cette petite modification m'a concrétisé ce fait que l'on nous répétait souvent en DUT ; que le client peut parfois faire des demandes incongrues et que ce sera **à nous de choisir entre réaliser la demande et suggérer une modification de manière professionnelle.**

Ce qu'elle fait

L'application contient maintenant une **petite barre de côté** contenant le bouton de sauvegarde, de remise à zéro (fait dans un excès de zèle), et les boutons d'incrémentation et

décrémentation de la valeur du Step que l'on veut changer. Une **barre d'onglet** sur le dessus nous indique de quel volume on est en train de modifier la configuration. **Les sliders de paramètres** sont toujours présent à l'identique grâce à une **section scrollable*** qui permet de continuer à mettre autant de paramètre que l'on veut tout en gardant nos boutons primordiaux cité au-dessus.

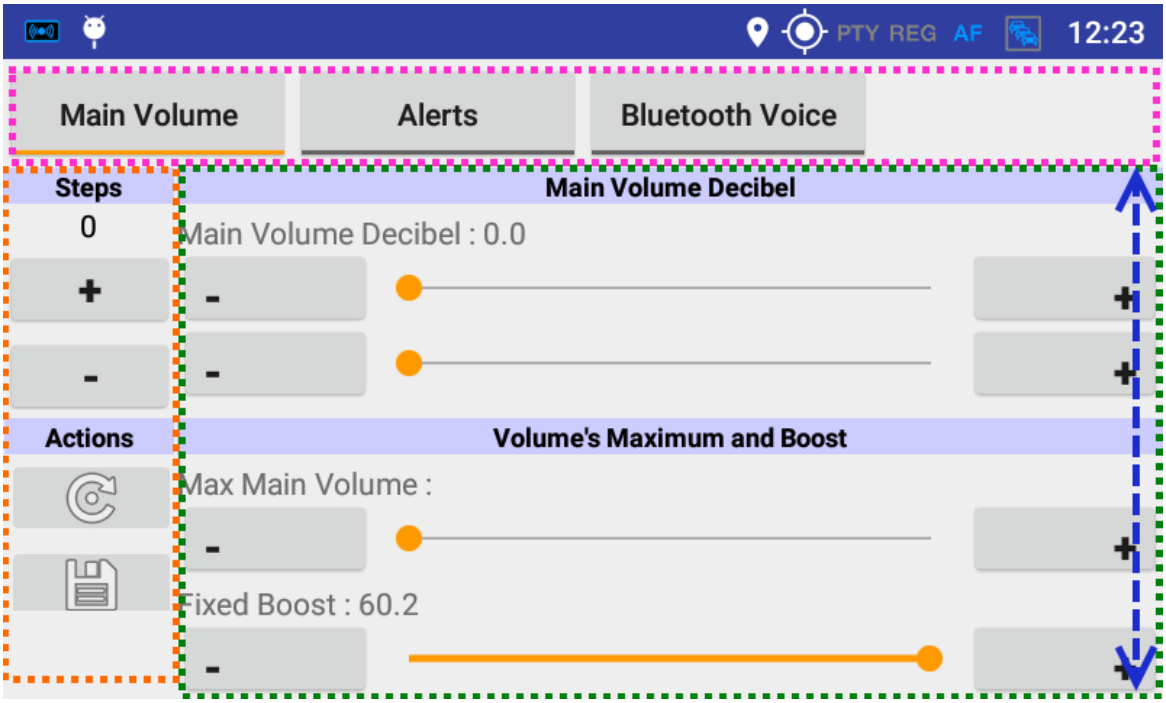


Figure 19 : Version 2 de l'application avec indications couleurs extrait

Comment elle le fait

Une fois encore, les préparations que j'avais faites ont été très utiles au moment de cette refonte graphique et à peine algorithmique. Les petits groupes des layout que j'avais créés se sont parfaitement intégré dans **leur section** spéciale. Section que j'ai pris le soin de rendre indépendante en cas de modification futures. Dans la version précédente **chaque sliders de chaque page** avait son action personnelle avec sa partie spécifique du xml qu'il modifierait et ceci attribué au chargement de l'application. Maintenant cette fonction de sauvegarde est plus groupée car ils ne modifieront que leurs tags xml associé dans la **section associée à l'onglet sélectionné.**

Cette refonte m'as également permis de purifier le code, de globaliser certaines variables et d'optimiser certaines fonctions. J'ai pu une nouvelle fois me sensibiliser à **l'importance de faire réviser son code**, même si cette fois c'était par moi. De plus, à chaque étape de l'amélioration que je passais était transmise sous forme de patch grâce à Git à mon chef. Il uploadait le patch sur Gerrit et attendait les retours des personnes à qui bénéficiaient l'application et d'une personne de l'entreprise qui faisait la review du code. Une fois tout le processus accomplit il me transmettait les retours. J'ai donc pu par l'intermédiaire d'A., participer et expérimenter le cycle de vie d'un patch et d'une correction de code.

Une mission complémentaire

En attendant que d'avoir une nouvelle tâche à me donner et pour rester dans l'optique de me faire apprendre le plus possible, A. m'a conseillé de me pencher sur **les tests automatiques** pour mon application. Cette fois-ci pas de consigne mais des conseils sur quel aspects tester. Il m'as conseillé de faire tests automatiques qui vont tester toutes les fonctionnalités de mon application et notamment les fonctionnalités ou série d'actions qui sont susceptible de faire crasher l'application quand j'y apporte une modification.

L'outil de test d'Android

Le DUT me l'as enseigné et j'ai pu le constater dans tous les aspects professionnels de l'informatique que j'ai rencontré : **Les tests, c'est important**. Il est donc tout à fait naturel que l'IDE par défaut d'Android bénéficiant d'un contrôle total d'un appareil (adb) ait **un outil dédié à des tests d'application.**

Cet outil est le Framework développé par Google : **UiAutomator.** C'est une bibliothèque comprenant tout ce qu'un développeur peut vouloir pour tester son application comme un utilisateur lambda l'utiliserait. Nous avions déjà vu ce principe en DUT avec les **tests unitaires** en C#. Il nous suffit de déclarer une classe, toujours en langage Java, qui dans sa définition contiendra les champs indiquant que cette classe est une classe de tests.

Pour indiquer que le fichier est un fichier de test on écrira à son début : `@RunWith(AndroidJUnit4.class)` et `@SdkSuppress(minSdkVersion = 18)`. Il faudra par la suite préciser avant chaque déclaration de fonction : `@Test` . Car il est possible de rédiger ses tests grâce à des fonctions indépendantes l'une de l'autre et ainsi pouvoir établir sa stratégie de test de manière très efficace en ordonnant, ignorant, rajoutant des fonctions qui testerons uniquement certaines fonctionnalité. Une fonction va alors utiliser toutes les méthodes qui lui sont disponible pour exécuter les actions désirés. Pour pouvoir exécuter complètement un test il faut 4 étapes dans son programme :

- **Chercher l'élément sur lequel on veut appliquer l'action**

C'est un élément très important et qui peut être effectué d'une multitude de façon. On a vu dans la partie **Comment est construite une application Android ? (p.14)** que la partie Ui (User Interface / Interface Utilisateur) était écrite dans un fichier xml contenant les éléments voulus ainsi que leurs ID si le développeur le précise. On peut donc utiliser une fonction très simple qui prend en paramètre le **type d'élément que l'on veut** (Bouton, section, slider, etc…) et **l'ID de l'élément**. C'est la façon la plus simple pour récupérer un élément. Il se trouve que dans la préparation aux modifications, j'avais attentivement Identifié chaque élément du layout de mon application. C'est donc cette méthode que j'ai utilisé le plus pour mes tests.

On peut aussi utiliser les **types d'élément** pour les trouver et utiliser leur **hiérarchie** (puisqu'il est tous encastrer les un sous les autres) ainsi on peut demander un élément de manière ordonné et en suivant les balises : le troisième bouton de la section sliders inclus dans la deuxième section de la troisième section de l'élément originel (l'application)

Si l'on n'a aucune information, on peut simplement sélectionner **une position**, sans aucun élément au bout. Et appliquer notre action sur le dit élément. Nous bénéficions donc d'une grande liberté de sélection pour tous les éléments possibles de l'application à tester.

- **Vérifier que l'élément est bien sélectionné**

Les tests automatique Android son quelque chose de très instable, Le but est justement de chercher à l'aveugle dans une application uniquement par rapport à son layout affiché, et d'exécuter des actions sur **ce qu'on pense** que l'affichage sera à ce moment T de l'application. Il faut donc continuellement faire des vérifications d'éléments pour éviter que le test ne crash au moment où il est juste en train de parcourir l'application afin d'aller à l'endroit critique du test. Il faut garder en tête qu'un test vérifie que l'application au bon comportement. Un test peut être réussit si une **application crash au moment voulu**. Mais un **test** de doit **jamais crasher tout seul** parce qu'il ne trouve pas tel ou tel élément.

Afin de préserver l'intégrité de la batterie de test durant leurs exécution, quand le développeur n'as plus aucune influence durant leurs exécution, il existe un outil très peu documenté appelé **UiWatcher.** C'est un objet qui est déclaré au début la fonction et qui va chercher certain éléments (à la manière classique d'un test) constamment durant l'exécution d'un test. S'il ne le trouve pas, tout est ok et le test se termine sans problème. Mais s'il trouve l'élément qu'il surveille, l'UiWatcher va mettre en pause l'exécution du test en cour, exécuter l'action pour l'laquelle il a été programmé puis reprendre le test là où il en était.

Ceci peut être très utile dans les grosses applications qui ne sont pas seul sur l'environnement ou elles sont testées. Par exemple une application qui se lance mais qui préviens l'utilisateur quand il n'a pas ou plus de connexion internet. Quand la batterie de test sera en train de tester une fonctionnalité n'ayant aucun rapport avec cette application et devra chercher un bouton, tout ce qu'elle trouvera sera le message d'erreur de l'application qui prévient de la coupure internet. **Si rien n'est fait, le test va crasher** et la batterie de test passera au suivant en marquant ce dernier comme défaillant alors que la fonctionnalité était parfaitement au point. Avec l'UiWatcher qui surveille constamment ce texte d'erreur spécifique, **le moment du crash sera invisible pour le test** et UiWatcher lancera l'action d'appuyer sur le bouton ok du petit message pour que l'application en cour de test soit de nouveau au premier plan. J'ai pu découvrir cet outil grâce à A. qui m'a dit à la fin de l'application de voir pour créer un UiWatcher.

La réalisation de l'UiWatcher n'a malheureusement pas aboutit à cause du fait que l'action qu'il voulait que l'UiWatcher exécute était déjà exécutée de base quand un l'application crashait durant un test. J'en ai juste tiré des connaissances sans mise ne application poussée.

- **Exécuter l'action sur l'élément critique**

Une fois que notre élément est existant et bien sélectionné, nous avons la possibilité de lui appliquer toutes les actions possible qu'un utilisateur pourrait faire avec son doigt. Pour UiAutomator tous les éléments sont les mêmes. Que ce soit un bouton ou une page entière, on peut y appliquer les actions d'appuyer sur le coin supérieur droit ou inferieur gauche, de

double appuyer, de glisser vers la droite. Il est donc possible de faire n'importe quelles actions depuis le coté de l'utilisateur final.

- **Vérifier que le comportement est conforme à la théorie**

Une fois toutes les actions effectuées, le but est de voir si le comportement attendu s'est produit ou non. Pour ce faire, UiAutomator nous met encore une fois plusieurs manières à disposition. La méthode la plus classique est d'utiliser la fonction `assertTrue(),` qui prends en paramètre le message à afficher dans la console en cas d'échec, et la condition à respecter. Si cette fonction renvois faux à un moment de l'application, le test se termine et déclare qu'il a échoué. C'est méthode peut être utilisé plusieurs fois dans une fonction.

La deuxième option souvent utilisé est de comparer une capture d'écran de l'appareil au moment de la vérification et de la comparer, pixel par pixel à une l'image de ce que l'appareil est censé afficher au moment voulu. Cette méthode peut avoir des gros avantage mais j'ai entendu dans le projet de Sébastien, mon collègue de l'IUT, que ceci était très contraignant car l'application ne faisait que changer du point de vue Interface Utilisateur.

Ce sont les principales fonctions et utilisation de UiAutomator proposé par Android que j'aie appris et les plus utilisé dans la programmation. Mais il faut prendre ne compte que beaucoup d'autres techniques de tests sont possibles et le fait que UiAutomator ne soit pas vraiment bien documenté augmente le nombre de possibilités inconnues que le Framework peut proposer. J'ai néanmoins pu utiliser et mettre en pratique ce que j'ai appris sans problème pendant cet entre-deux missions.

La première amélioration : une nouvelle fonctionnalité

L'application s'étant parfaitement développée et m'étant occupé des tests de manière assez complète. Mon chef d'équipe a décidé que je pouvais passer à la partie suivante qui était de rajouter une fonctionnalité à l'application.

La nouvelle fonctionnalité

La nouvelle fonctionnalité à ajouter n'était pas du tout complexe à intégrer pour la simple et bonne raison qu'elle consistait à pouvoir, avec des sliders, modifier précisément dans un fichier xml le comportement de **l'amplification sonore (dit loudness)** du volume. Le sujet m'a été présenté de cette façon :

Ok, now we need the following new sliders that will change another xml file when we press on save:

6 new sliders
1 for MaxBassBoost, from 0 to 19, default is 19
1 for MaxTrebleBoost, from 0 to 14, default is 14
2 for BassBoost
1 for integer part from 0 to MaxBassBoost
1 for decimal part from 0 to 99
2 for trebleBoost
1 for integer part from 0 to MaxTrebleBoost
1 for decimal part from 0 to 99

and a loudness on/off button that should always be present maybe if it can be put on the left side of the screen under the volume.

Par la suite, A. m'as conseillé sur Skype d'intégrer cette fonctionnalité dans un nouvel onglet du même style que les trois précédents de telle sorte à ce qu'à la fin, l'application affiche : Main Volume, Alert, Bt_Voice et Loudness.

Les solutions et améliorations techniques

On peut se dire de prime abord qu'avec la ressemblance de la consigne précédente, il m'a été facile de compléter l'application avec cette fonctionnalité. Mais l'objectif de la programmation (quand ce n'est pas à 100% l'optimisation) est de faire un code modulaire pour prévoir les modifications et ne jamais devenir obsolète. Et dans cette situation, ma fonction **d'optimisation** qui sauvegardait tout de la même façon (en parcourant les sliders en sauvegardant leurs valeurs le xml) par rapport à l'onglet sélectionné **m'empêchait de moduler** mon code pour intégrer facilement cette fonctionnalité.

En effet, s'il avait s'agit de rajouter un onglet avec le même fichier xml (et donc les même sliders) cela aurait été très simple puisque je n'aurais eu qu'à rajouter l'onglet et sa balise associer dans le programme et la fonction de sauvegarde aurait fonctionné de la même façon.

Seulement, ici le **fichier xml est différent et les sliders aussi**. Si j'avais été un mauvais programmeur, j'aurais juste fait un cas spécial qui sélectionnerait d'autre sliders pour la sauvegarde et au final récrire toute les fonctions de sauvegarde dans ce cas spécial. Et de même pour l'affichage des sliders. Car je n'avais plus besoin de charger 3 fois une page différentes avec l'exact même contenu, mais juste laisser une section de sliders qui ne bougent pas et ne servent qu'à prendre les valeurs du Step (dans la barre verticale demandée) voulu et l'onglet sélectionné.

J'ai donc dû rendre modulaire le fonctionnement de la section sliders de mon application. Pour ce faire j'ai dû réfléchir à ce que je voulais exactement : Je voulais que peu importe l'onglet sélectionné, j'appelle une fonction sauvegarde qui sauvegarderait les valeurs de sliders **affichés**. Je voulais aussi pouvoir charger ces sliders avec les valeurs qui leurs correspondaient. Il me fallait donc un **fragment** contenant ayant les mêmes **méthodes** mais que ces méthodes n'aient pas le même **comportement**.

C'est exactement le principe d'une **interface**. Une interface va être un ensemble de fonction que l'on peut utiliser (dans notre cas charger et sauvegarder) et qu'on va relier à un objet qui peut implémenter* cette interface. J'ai donc modifié mon code de la façon suivante :

J'ai commencé par vider la section de sliders et créer deux layouts *[Voir Annexe 6 : fragments de l'application finale]* contenant les sliders appropriés. Je me suis permit de proposer une amélioration de l'application en précisant qu'avoir un slider pour le max deux fois de suite était peut-être un peu chargé puisque ça allait obligé l'utilisateur à faire descendre l'application pour modifier la valeur maximum du slider au-dessus. A. en a parlé aux intéressés et m'a transmis leurs accord. J'ai donc changé les sliders de maximum par deux boutons supplémentaires. Cela m'as montré que malgré ne consigne précise, on peut toujours **essayer d'améliorer** notre produit et prendre des initiatives, à condition que cela soit approuvé par les clients.

La section de sliders n'est maintenant plus qu'un **réceptacle à ces layout**. Maintenant que j'en avais acquis les compétences, j'ai pu créer deux classes de fragment héritant de la superclasse Fragment. C'est exactement le même principe, la même astuce, qu'avait utilisé Android à la création du modèle de notre application. Seulement ici ce sont réellement des fragments de layout, et non pas des pages entières. Une classe `FragmentVolume` et une classe `FragmentLoudness` ont donc été créée.

Ces classes ont au moins deux définition de fonction en commun : `saveValues()` et `loadValues()`. Ce qui fait qu'on va pouvoir créer notre interface : `I_SettingFragment`. Ce fragment va donc contenir seulement la définition de nos deux fonctions précédemment citées. La classe MainActivity (le MainActivity.java vu dans la partie "Comment est conçue une application Android") a été dépouillée de ses fonctions de chargement de sliders et de sauvegarde. Dorénavant la classe MainActivity, l'application donc, ne fera qu'appeler les fonction saveValues() ou loadValues() du **Fragment courant**. Ce fragment courant sera chargé dans l'espace qui lui est dédié précédemment (la section qui contenait précédemment les sliders). Tout ce qu'on déclare dans le programme est le fait que une fonction saveValues() et loadValues() seront toujours disponible au moment où il faudra les utiliser, mais le fragment qui exécutera ces fonctions ne seront pas toujours les mêmes.

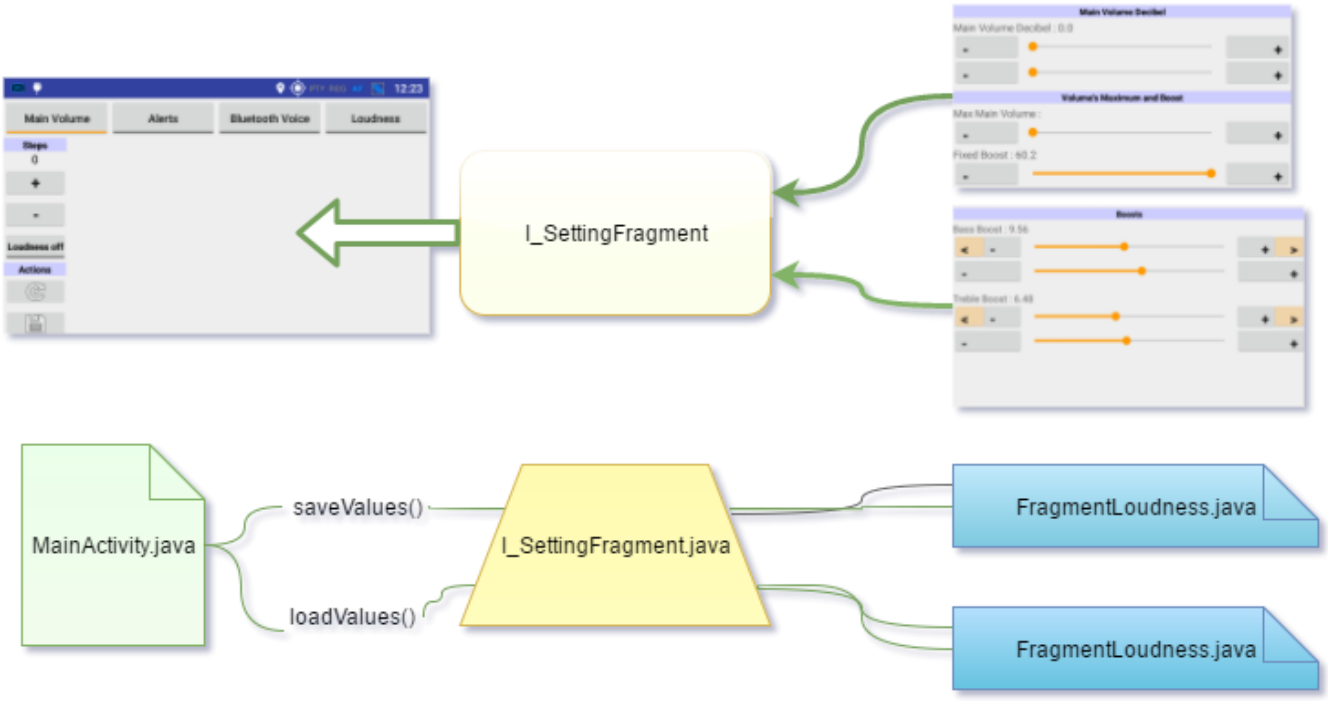


Figure 20 : Schéma représentatif du fonctionnement de l'interface I_SettingFragment au sein de l'application avec au-dessus : la partie graphique (layouts) et en dessous : la partie programme (classes)

Nous avons donc notre modularité, nous pouvons créer autant de sliders de paramétrage désiré, il suffira que la classe qui contient leurs fonctionnement implémente l'interface I_SettingFragment et que l'action de charger ce type de fragment au moment ou tel onglet est cliqué soit programmé. À partir de ce point, nous pouvons rajouter autant de type de modification de son et de fichier que l'on veut et facilement.

J'ai pu, grâce au test automatisé que j'avais précédemment conçus et amélioré pour cette version de l'application**, grandement faciliter mes moments de tests de l'application**. Je pouvais travailler un peu sur mon rapport quand les tests tournaient ou simplement prendre une petite pause… Non pas qu'il me fallait une excuse pour prendre un pause, J'ai déjà expliqué dans la partie **Culture d'entreprise (p.4)** que les pauses peuvent être prise quand bon me semble, mais disons que je ne perdais pas de productivité quand j'allais profiter du soleil dans la cour.

Encore une fois, j'ai clairement appris de cette expérience et l'appliquerais dès que possible dans ma vie professionnelle ou mes projets personnels futurs.

L'environnement culturel

Comme je l'ai mentionné à différents endroits de ce rapport, j'ai eu la chance d'effectuer mon stage en Roumanie. Dans un contexte culturel dont je n'avais aucune informations ou celles que j'arrivais à obtenir de personne ayant déjà eu l'occasion d'y aller se contredisait. Le fait de devoir apprendre en plus d'une culture d'entreprise, la culture d'une nation m'a grandement ouvert aux changements possibles que le fait de travailler dans une nouvelle

entreprise peut apporter. Mais aussi le changement que le fait de travailler à l'étranger peut apporter.

Une attention particulière était continuellement apportée à l'opinion des gens sur leurs propre pays et la façon de voir la cohabitation, ou simplement la façon de vivre d'un individu. Surtout dans un pays aussi décalé de ce dont j'avais l'habitude de par sa sortie du communisme des années auparavant. Ici les mentalités était devenu beaucoup plus proche du chacun pour soi dans tous les petits aspects de la vie quotidienne, à l'opposé du communisme donc, que ce que l'on peut voir en France, même dans des grandes villes réputée avec une population peu accueillante.

Et c'est également pour ça que l'entreprise dans laquelle j'étais était aussi agréable et avenante. L'américanisation de la culture d'entreprise étant encore plus contrasté au milieu de cadre postcommunisme on en arrivait à des efforts de bienveillance inimaginable en France.

J'ai donc pu être sensibilisé à cette notion de travail mérité et la différence entre ce qu'on peut réussir à accomplir au milieu d'un cadre en apparence non-prometteur. J'ai pu voir aussi comment, après une cassure avec le monde extérieur, une population pouvait se suffire à elle-même avec ses codes et ses règles implicites de bienséances. Toutes ces choses que l'on prend l'habitude de considéré comme acquis et inaliénable et pourrait nous rendre plus difficile une collaboration avec d'autre personne qui ne serait pas du même milieu ou de la même culture. Cela a vraiment été enrichissant de pouvoir vivre une expérience de déracinement aussi profonde de ces habitudes qui nous rendent moins flexible et moins ouvert.

Grace à la diversité culturelle de l'entreprise, j'ai pu collecter des expériences d'autres employés très intéressantes comme des programmeurs Senior tout comme Junior qui me racontait leur vision et leurs expériences avec tel ou tel langage et qui me conseillait de plutôt privilégier tels ou tels principe de vie. J'ai aussi eu le point de vue de plusieurs employés sur ce qu'ils pensaient de la méthode AGILE dont on nous parle autant depuis deux ans au DUT. Selon eux, c'est surtout un style de communisme informatique qui ne peut que s'empirer.

Au-delà de ça, j'ai pu observer une volonté et une adaptabilité face aux difficultés de la vie que je n'avais jamais observée chez une si grande communauté auparavant. J'ai également observé un mélange de classes sociales qui serait inacceptable en France, on pouvait croiser n'importe quel type de personnes, de la plus défavorisé à la plus aisé dans les mêmes rues de la ville. Ceci à cause de la reconstruction éparse qui s'est effectué après la guerre, les points d'intérêts ne se sont pas regroupé comme ils le feraient dans une vielle ville mais ils sont localisé à des endroits parfois contre instinctifs ou au milieu de quartiers qui n'ont pas l'air d'avoir bénéficié de beaucoup de rénovations

Au final ces expériences à l'étranger m'ont appris des principes et apporté une autre vision de la vie que je n'aurais pas pu appréhender si je n'avais pas fait ce stage à l'étranger.

Résultats, Interprétations, Bilans

Résultats factuels

Avec du recul, mon stage ressemblait plus à un stage d'observation très actif, qu'à un stage au sein d'une entreprise qui avait besoin de moi. Cela m'a donc valu d'apprendre beaucoup plus que ce que j'ai apporté à l'entreprise. Je n'ai réalisé qu'une application destiné aux clients/revendeur en tant qu'outils et rédigé des tests sur cette même application. C'est bien peu de chose pour deux mois et demi de stage.

J'ai par conséquent utilisé ce temps pour apprendre une très grande quantité de choses sur un nombre important d'aspects de mon entreprise d'accueil. J'ai appris comment fonctionnait une entreprise qui favorisait ses employés, comment un projet de grande envergure pouvait avancer et comment le vivait ses acteurs.

J'ai appris à développer dans un nouvel environnement en développant à fond de manière autonome mes connaissances, sans contraintes de temps ni de pression extérieure. J'ai appris une nouvelle culture et observé aussi bien que participé à l'aspect social du bâtiment dans lequel j'ai fait mon stage.

Dans un ton plus personnel et social j'ai aussi beaucoup interagit avec mes collègues et appris de leurs expériences personnelles. J'ai participé à différents évènement sociaux de l'entreprise (match d'ouverture de l'Euro de football, fête mensuelle des événements de l'entreprise, festin improvisé pour une étapes importante passé dans le projet, etc…) et ai pu me construire une image de l'entreprise et mon pays d'accueil très développée grâce à toutes expériences et discutions que j'ai pu avoir avec des personnes qui n'avait jamais 2 choses en commun.

Sens de ses résultats par rapport à l'objectif

Mon objectif au départ était d'en apprendre le plus possible sur comment fonctionne une entreprise d'envergure non négligeable. Il était aussi de recevoir le plus d'expérience possible venant du changement culturel et de mes collègues, en tant que personnes et non employés. Toutes les informations concrètes, sous-entendu non-sociales ou non-culturelles, que j'ai pu accumuler l'ont été car je m'étais fait des contacts dans le cadre de la création mon application (chef d'équipe, secrétaire en chef, membres de l'équipe etc…) et cette application a finalement été ma passerelle avec toutes les connaissances que j'ai pu avoir dans l'entreprise. Puis à coter de cette expérience d'entreprise, j'ai pu recevoir cette expérience sociale et culturelle qui ne peut être décrite qu'approximativement avec des mots.

Si l'on compare ces résultats que j'ai décrit précédemment cité comme objectif et cette question directrice :

« Comment fonctionne une entreprise de dimension internationale et comment y vivent ses employés ? Comment se déroule un projet au sein de cette entreprise ? Que retirer d'une telle expérience à l'étranger ? »

J'estime avoir honoré en très grande partie mon objectif. J'ai pu rassembler quantité d'informations et d'expérience sur le maximum de domaine que mon cadre professionnel me le permettait. L'aspect culturel n'as pas été négligé le moins du monde. Avec un décalage aussi important jusqu'à même mon appartement, je n'ai cessé, dès que j'ai posé le pied en Roumanie, de découvrir de nouveaux aspects de la ville, du mode de vie de la population autochtone ou des relations humaines en entreprises

. Je garde une réserve cependant quand à mon expérience en entreprise. Car l'entreprise étant idyllique dans tout ce qu'elle propose à ses employés et son organisation, je n'ai pas eu une expérience complète d'une entreprise moyenne française. J'ai évidemment communiqué avec mes autres camarades, et j'ai pu constater de la grande différence qu'il y avait entre ce que mes amis vivaient en entreprise, et les projets qu'ils avaient à réaliser et mon stage d'observation amélioré. Je ne me sens donc pas aussi expérimenté qu'eux si je devais rentrer dans entreprise française dans les mois qui suivent ce stage. Malgré cela, part le prisme de leurs ressentit, j'ai pu avoir une petite expérience des désagréments possibles, des imprévus et des attentes que des entreprises plus conventionnelles française peuvent avoir d'un employé en informatique.

Selon moi, le résultat de ce stage est à la hauteur de ce que l'IUT veut nous faire vivre. Seulement pas dans les même domaines. Si j'avais été dans cette même entreprise implantée en France, je n'aurais certainement pas tiré autant d'expérience, et en ce sens, le stage n'aurait pas été aussi réussi qu'il le fut.

Bilan Professionnel

Du point de vue purement professionnel, ce stage me sera beaucoup plus utile et pertinent pour m'intégrer dans une grande compagnie tel que WindRiver plutôt que pour travailler dans le service informatique d'une entreprise moyenne quelconque.

En partie car j'ai surtout développé mon sens du collectif et ma vision de l'organisation d'un projet informatique de grande envergure. J'ai observé des mécaniques de synchronisation et d'organisation propre aux entreprises américanisées précédemment évoqué. L'organisation des équipes et cette hiérarchie qui permet une résolution des problèmes efficaces est parfaite pour travailler en équipe mais pas pour un employé seul ou en équipe de 4 qui doivent s'occuper de mettre en place et maintenir toute la plateforme informatique entière de l'entreprise (comme certain de mes camarades à Craïova). Je ne peux pas vraiment dire que mon sens des responsabilités a été mis en pratique car mon application ne représentait aucun enjeu pour l'entreprise ou l'équipe. Si je n'avais pas pu la réaliser, un autre membre de l'équipe s'en serait chargé plus rapidement. **Ce manque de responsabilités à cause de la découverte constante de l'entreprise et du pays est l'aspect le plus négatif qui ressort de ce stage.** Je ne pouvais pas arriver dans un projet bien en marche sans aucune connaissances ni de l'entreprise ni du projet. Cela est dû aussi beaucoup au fait que j'ai été intégré dans cette entreprise sans avoir le moindre choix et surtout avec des informations succinctes arrivés à l'absolu dernier moment.

Cependant, mon expérience de rendu étape par étape pour montrer au client mon parcours et le rapport systématique me sera cependant utile dans tous les domaines. La

review de code, la correction par rapport à des exigences ainsi que la possibilité de prendre des initiatives et proposer des améliorations m'ont initié à la participation d'une équipe pour un projet d'envergure. Et l'adaptabilité demandée sera toujours présente lors d'une prochaine mission de plus grande ampleur. Mais c'est à peu près tout ce qui m'a été inculqué car le reste n'aura été que de l'initiation.

Il est important de préciser que c'était la première fois, depuis que j'apprends l'anglais, que j'ai pu pleinement le pratiquer et que j'ai dû compter sur ma maitrise de la langue pour vivre et interagir avec mon environnement professionnel ou personnel. Cela a été indéniablement une expérience enrichissante et utile pour va vie professionnelle future.

Bilan Personnel

J'estime que mon expérience personnelle a été autant, sinon plus importante que mon expérience professionnelle.

Comme expliqué dans la partie Environnement culturel j'ai vécu une quantité incroyable d'interactions sociales que je n'avais jamais vécues auparavant. J'ai vu et entendu des choses qui m'ont fait réfléchir d'une façon différente de celle dont j'ai l'habitude. Cette découverte d'une nouvelle société a beaucoup apporté à mon enrichissement personnel. En partie car on distingue vraiment une identité typiquement roumaine sans influence extérieure. Ceci à cause des fréquentes occupations qu'a subit le pays jusqu'à très tard dans leur histoire.

Les relations que j'ai établies avec mes collègues ont été également enrichissantes. Les événements sociaux et lieu de rassemblement et d'échanges de WindRivier (ping pong, babyfoot, fêtes, cafétéria agréable à vivre, etc..) aussi bien que de la ville (pub, parc, etc..) m'ont permis de lier des relations de sympathie et d'échange avec beaucoup d'individus.

Mais plus intéressant encore, c'est la relations que j'ai vécu avec mon collègue, camarade de classe, compagnon de voyage et colocataire Sebastien Behr. Avant notre départ pour la Roumanie, nous n'avions pas particulièrement de bon sentiments l'un envers l'autre. Mais à travers notre constante et commune découverte de la Roumanie et de WindRiver, nous avons sympathisé, échangé nos points de vue, discuté de sujets divers et varié et partagé nos expérience pour un résultat que je n'avais définitivement pas prévu. Nous n'avons pas lié une amitié à proprement parler mais plutôt une entente amicale qui m'a fait passer un séjour très agréable et formateur.

Cette expérience à l'étranger m'aura ouvert l'esprit quant aux possibilités et surprises que notre société et notre monde contiennent et qui n'attendent que de l'initiative et de l'ouverture d'esprit pour être découvert.

Conclusion

La difficulté la plus évidente que je pourrais citer était bien évidement la barrière de la langue. Socialement comme professionnellement, il a fallu se reprendre à plusieurs fois pour expliquer un principe ou une logique abstraite. Il fallait que l'émetteur soit assez concentré pour ne pas faire de faute de grammaire majeure, que le receveur soit assez concentré pour comprendre correctement le message et enfin que les deux aient à peu près la même chose en tête pour ne pas trop s'écarter et interpréter d'une autre façon le message.

Mis à part ceci, il est plus facile de noter la non-absence de difficulté qui a rendu ce stage si différent de ce que la majorité des élèves de ma promotion ont vécu. Il a donc fallu que la recherche de connaissance et l'apprentissage de compétences se fasse de ma propre initiative. Ceci a été une difficulté qui a découlé du manque de difficulté apparent de mon stage. Cette difficulté n'existait cependant que par ma soif de connaissance et c'est peut-être une des difficultés les plus bénéfiques et gratifiantes que j'ai rencontré depuis que j'apprends l'informatique.

Grace à l'initiative dont j'ai fait preuve et l'aide de mes collègues, j'ai pu m'initier à un nouvel IDE, me parfaire en Java, découvrir l'environnement Android et observer le fonctionnement, et les employés influencé par un mangement à l'américaine. Mais j'ai aussi pu m'enrichir par l'immersion dans une nouvelle culture totalement étrangère à la mienne et rythmé de rencontres inédites. La richesse de ces échanges a été une occasion de réaliser un travail sur moi qui m'est aujourd'hui profitable dans de multiples domaines et qui le sera certainement dans l'avenir.

Si je devais recommencer en ayant conservé l'expérience que j'ai acquéris à la fin de ce stage, au tout début de mon entretiens avec le PM du projet, j'aurais précisé que je préfèrerais apprendre à faire des tests automatisé plus complexes afin d'être utile à l'équipe. Je pense aussi que je ferais de mon mieux pour faire quelque chose de plus productif et avantageux pour mes collègues dans mes périodes de vide. Au final, je pense que je demanderais simplement plus de responsabilités.

Je pense que mon stage, aussi intéressant fut-il, n'avait que peu de choses en commun avec ce qu'un élève est censé faire de ses deux mois et demi en fin de DUT. Pourtant, c'est ce qui le rend aussi exceptionnel et enrichissant pour la suite de ma vie professionnelle et personnelle.

Glossaire

API : Interface de Programmation Applicative – Ensembles de services (Classes*, fonctions, méthodes) mis à la disposition d'autre logiciels quelconques par un logiciel en ligne ou local.

Board : Anglicisme désignant toute la partie physique du tableau de bords. Peut-être appelé carte de contrôle.

Boucle While : Dans du code - Encapsulation simple dans laquelle on code une ou plusieurs actions à effectuer et la condition de répétition de ces actions.

Build : Ensemble de fichiers contenant un projet complet prêt à être utilisé par d'autres programmes.

Classe : Dans l'Informatique – Fichier contenant toute les fonctions et variables spécifiques à cette classe. D'autre classe pourront utiliser les variables et fonctions rendues publiques d'une classe au sein du même programme.

Client : Dans l'Informatique – Logiciel qui reçois les informations depuis un système de communication vers d'autres machines, logiciels ou programmes présents sur un réseau informatique (privé ou publique). Ce client est local et chez la machine de l'utilisateur. Il peut désigner une invite de commande, un navigateur web (client web), etc…

Compilateur : Programme Informatique qui va interpréter le code écrit dans un langage donné pour le stocker sous forme d'instructions compréhensible pour un microprocesseur.

Couche : Dans l'Informatique – niveau d'abstraction du code, une couche spérieure utilisera des fonctions de la couche du niveau inférieur pour fonctionner et faire abstraction des aspects physiques et élémentaire de la communication avec les composants. Plus une couche est basse, plus elle est proche des composant tandis que plus une couche est haute, plus elle sera plus simple à utiliser mais avec des boites noire dans le code.

CPU : Computing Processing Unit / Unité Centrale de Traitement – Composant électronique capable d'exécuter des opérations logiques plus ou moins complexes sur commande. Elément indispensable à tout système de traitement informatique.

Daemon : Processus ou ensemble de processus s'occupant de tâches légères, principalement de la communication, et fonctionnant en boucle sans avoir d'impact important sur la charge de travail du processeur.

Deadline : Anglicisme pour la date limite du rendu final d'un projet.

Driver : Programme permettant la communication entre un système d'exploitation et un composant électronique spécifique au driver.

Émulateur : Logiciel permettant de simuler au sein d'un système d'exploitation l'ensemble du fonctionnement physique comme logiciel d'un autre système Informatique.

Encapsulation par try catch : Dans du code - Encapsulation simple dans laquelle on précise d'une part une ou plusieurs actions à exécuter et d'autre part le type d'erreur qui peut se produire pendant cette action et une ou plusieurs actions à exécuter si cette erreur se produit.

Exception : Une exception est un objet* représentant un bug survenu lors de l'exécution d'un programme. Il est possible de l'empêcher d'apparaitre et donc d'empêcher le programme de crasher grace à une encapsulation try catch*

Framework : Ensemble cohérent de composants logiciels génériques servant de base à un logiciel, ou de complément à une application

Hardware : Anglicisme pour designer tous les composants électroniques physiques d'un système informatique.

Hériter : Dans l'Informatique – Une Classe* hérite d'un autre quand elle veut copier l'exact fonctionnement de l'autre (en récupérant ses fonctions et variables accessible par héritage) tout en se développant autour de cette base acquise.

IDE : Interactive Development Environment / Environnement de Développement Interactif – Logiciel de développement possédant des caractéristique aidant à la programmation d'un ou plusieurs langages.

Implémenter : Action d'une Classe* de remplir toutes les conditions pour pouvoir être utilisée à travers une interface (Cad, posséder toutes les fonctions déclarées dans l'interface).

Internet of Things : Se dit de la globalité des appareils connecté, petits ou gros, à un réseau avec un accès direct ou indirect à Internet. Cette masse monumentale d'appareils connectés ensembles contiennent une quantité tellement massive de données sous diverses formes que à ce jour, personne n'as réussi à faire un traitement satisfaisant de l'entièreté de ces données.

Kernel : Nom donné au cœur du fonctionnement de Linux. Ce noyau dur de fonctions et services font la base ultime de tout systèmes d'exploitations dérivants de Linux.

Mémoire : Dans l'Informatique – Composant électronique capable de stocker des données pendant une très longue durée. Cette mémoire peut nécessiter d'être constamment alimenté ou non.

Machine virtuelle : Sur le principe de l'émulateur*, Logiciel simulant le fonctionnement complet d'un système d'exploitation avec ses composants électroniques. Illusion d'un appareil informatique créée par un logiciel d'émulation.

Méthode : Dans l'Informatique – Fonction d'une Classe*.

Network Function Virtualization : Utilisé dans la simulation informatique, principe permettant de simuler des réseaux informatiques à grande échelle sans en avoir le matériel.

Objet : Classe qui est utilisé au sein d'un programme comme une variable, on pourra appeler et utiliser les méthodes* et variables* de cette classe à travers la variable créée appelée Objet.

Propriétés : Dans l'Informatique – Variable d'une Classe*.

Scrollable : Anglicisme pour le fait qu'un affichage est plus grand que l'écran d'affichage et que l'on peut le parcourir dans sa longueur ou largeur par une action définie.

Serveur : Dans l'informatique – Logiciel communiquant avec des clients* et autre serveurs non-utilisé directement par les utilisateurs. Il s'occupe de faire des actions privés, du stockage ou de la redirection.

Software : Anglicisme pour Logiciel, par opposition au Hardware*.

Sliders : Anglicisme pour curseur de défilement, c' est un composant d'interface graphique permettant d'entrer une valeur numérique dans un programme en déplaçant un curseur sur une échelle graduée.

Systèmes embarqués : Système informatique autonome comprenant tout le Hardware* et le Software*.

Annexes

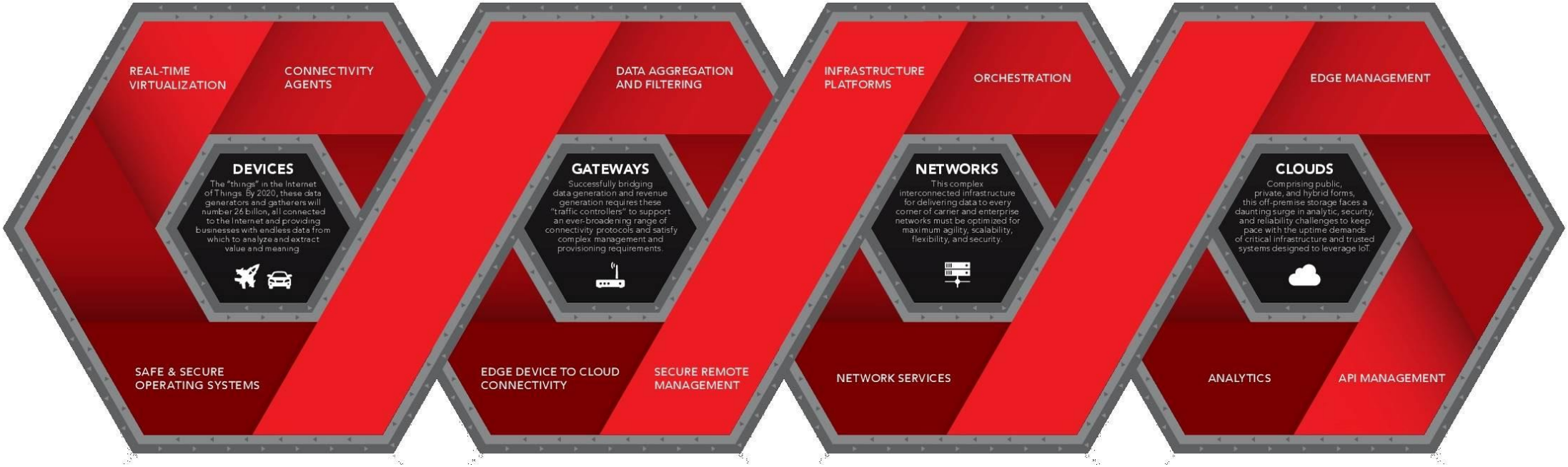


Figure 21 : Représentation de la politique de développement technologique de WindRiver concernant l'IoT

Annexe 1 : WindRiver Helix [13]

Annexe 2 : Locaux de l'entreprise

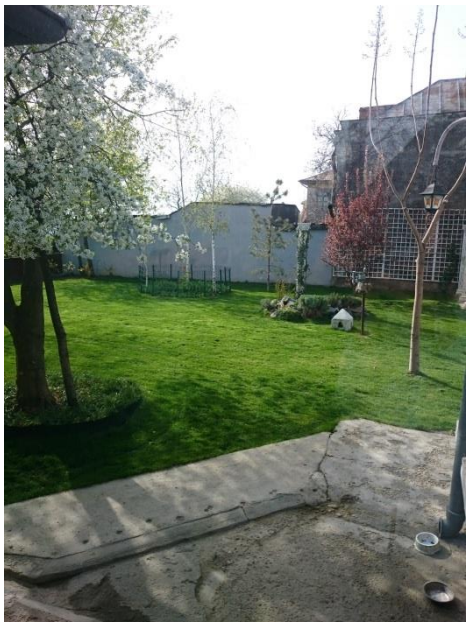


Figure 25 : Jardin de WR3

Figure 24 : Salle à manger de WR3

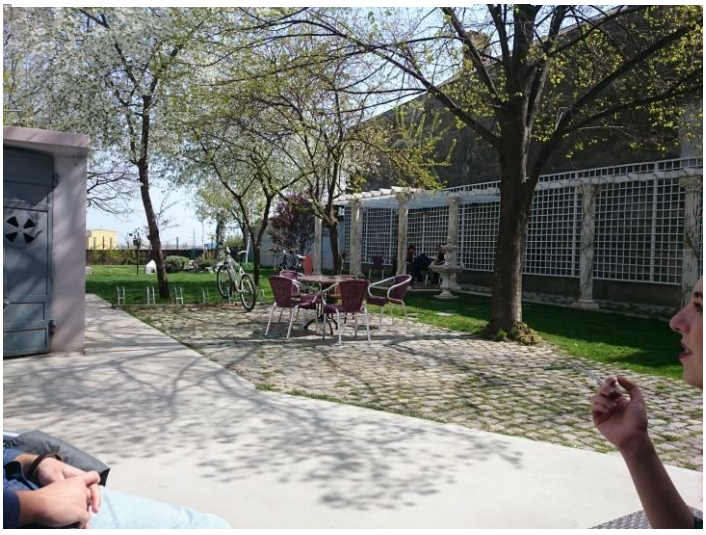


Figure 23 : Jardin de WR3

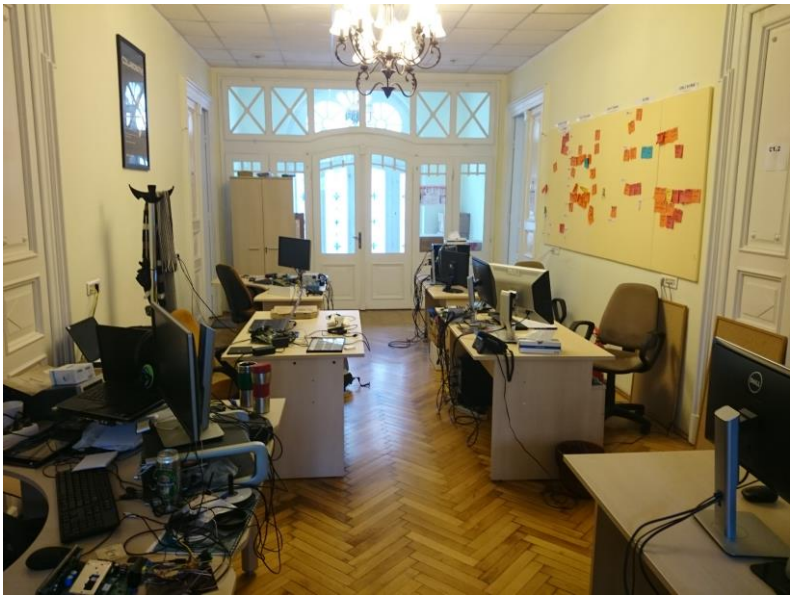


Figure 22 : Un des bureaux open-space de WR3

Annexe 3 : Extrait du document XML des valeur de volumes

```
<?xml version="1.0" encoding="UTF-8"?>
<VolumeConfig>
    <Defaults>
        <!--<nr_of_steps>10</nr_of_steps>-->
        <FixedBoost>48.16</FixedBoost>
        <MainVolMax>12.04</MainVolMax>
        <MaxLoudBoost>19.0</MaxLoudBoost>
        <VoldB>0.0</VoldB>
    </Defaults>
    <Stream id="main" channels="2" >
        <Main1Address>0xF44084</Main1Address>
        <Main2Address>0xF44085</Main2Address>
        <Steps>
            <nr_of_steps>33</nr_of_steps>
            <Step id='0'   VoldB='-50' FixedBoost='48.16' MainVolMax='12.04' MaxLoudBoost='19.(
            <Step id='1'   VoldB='-48.4375'></Step>
            <Step id='2'   VoldB='-46.875'></Step>
            <Step id='3'   VoldB='-45.3125'></Step>
            <Step id='4'   VoldB='-43.75'></Step>
            <Step id='5'   VoldB='-42.1875'></Step>
            <Step id='6'   VoldB='-40.625'></Step>
            <Step id='7'   VoldB='-39.0625'></Step>
            <Step id='8'   VoldB='-37.5'></Step>
            <Step id='9'   VoldB='-35.9375'></Step>
            <Step id='10'   VoldB='-34.375'></Step>
            <Step id='11'   VoldB='-32.8125'></Step>
            <Step id='12'   VoldB='-31.25'></Step>
            <Step id='13'   VoldB='-29.6875'></Step>
            <Step id='14'   VoldB='-28.125'></Step>
            <Step id='15'   VoldB='-26.5625'></Step>
            <Step id='16'   VoldB='-25'></Step>
            <Step id='17'   VoldB='-23.4375'></Step>
            <Step id='18'   VoldB='-21.875'></Step>
            <Step id='19'   VoldB='-20.3125'></Step>
            <Step id='20'   VoldB='-18.75'></Step>
            <Step id='21'   VoldB='-17.1875'></Step>
            <Step id='22'   VoldB='-15.625'></Step>
            <Step id='23'   VoldB='-14.0625'></Step>
            <Step id='24'   VoldB='-12.5'></Step>
            <Step id='25'   VoldB='-10.9375'></Step>
            <Step id='26'   VoldB='-9.375'></Step>
            <Step id='27'   VoldB='-7.8125'></Step>
            <Step id='28'   VoldB='-6.25'></Step>
            <Step id='29'   VoldB='-4.6875'></Step>
            <Step id='30'   VoldB='-3.125'></Step>
            <Step id='31'   VoldB='-1.5625'></Step>
            <Step id='32'   VoldB='0'></Step>
        </Steps>
    </Stream>
    <Stream id="alert" channels="1" >
        <Main1Address>0xF440BD</Main1Address>
        <Steps>
            <nr_of_steps>33</nr_of_steps>
            <Step id='0'   VoldB='-33'></Step>
            <Step id='1'   VoldB='-32'></Step>
            <Step id='2'   VoldB='-31'></Step>
            <Step id='3'   VoldB='-30'></Step>
            <Step id='4'   VoldB='-29'></Step>
            <Step id='5'   VoldB='-28'></Step>
            <Step id='6'   VoldB='-27'></Step>
            <Step id='7'   VoldB='-26'></Step>
            <Step id='8'   VoldB='-25'></Step>
            <Step id='9'   VoldB='-24'></Step>
            <Step id='10'  VoldB='-23'></Step>
            <Step id='11'  VoldB='-22'></Step>
            <Step id='12'  VoldB='-21'></Step>
            <Step id='13'  VoldB='-20'></Step>
            <Step id='14'  VoldB='-19'></Step>
            <Step id='15'  VoldB='-18'></Step>
            <Step id='16'  VoldB='-17'></Step>
            <Step id='17'  VoldB='-16'></Step>
            <Step id='18'  VoldB='-15'></Step>
            <Step id='19'  VoldB='-14'></Step>
            <Step id='20'  VoldB='-13'></Step>
            <Step id='21'  VoldB='-12'></Step>
            <Step id='22'  VoldB='-11'></Step>
            <Step id='23'  VoldB='-10'></Step>
```

Figure 26 : Extrait de document XML de valeurs

Annexe 4 : Capture d'écran de l'application une fois le design visuel terminé

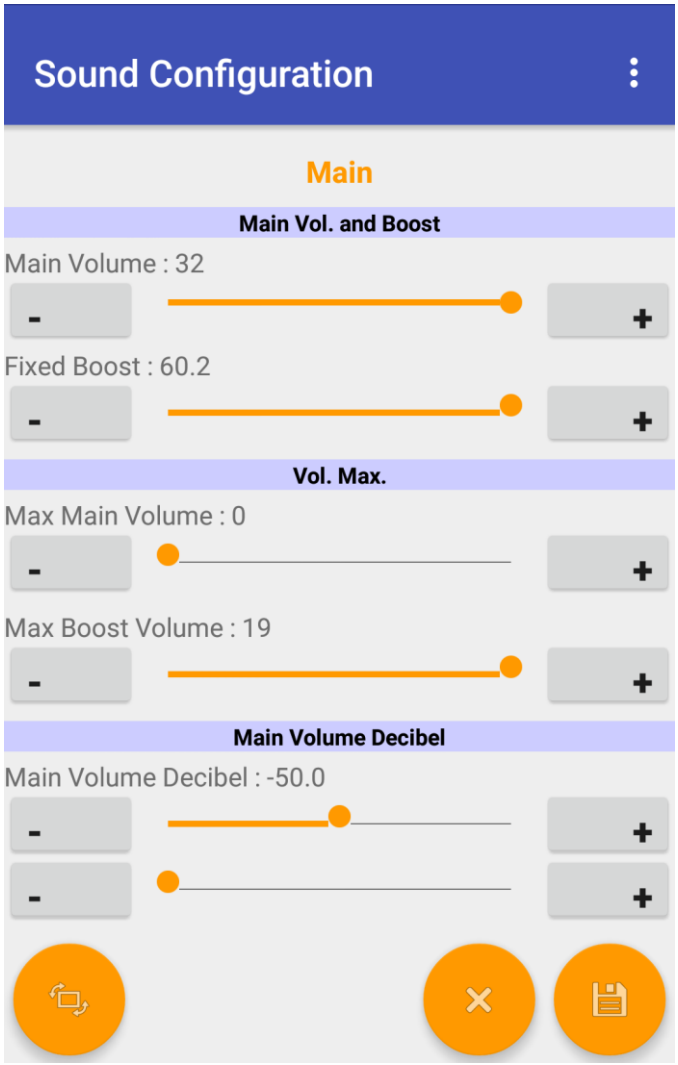


Figure 27 : Capture d'écran de l'application une fois le design visuel terminé

Annexe 5 : Schéma représentatif du workflow d'un patch dans Gerrit selon Android

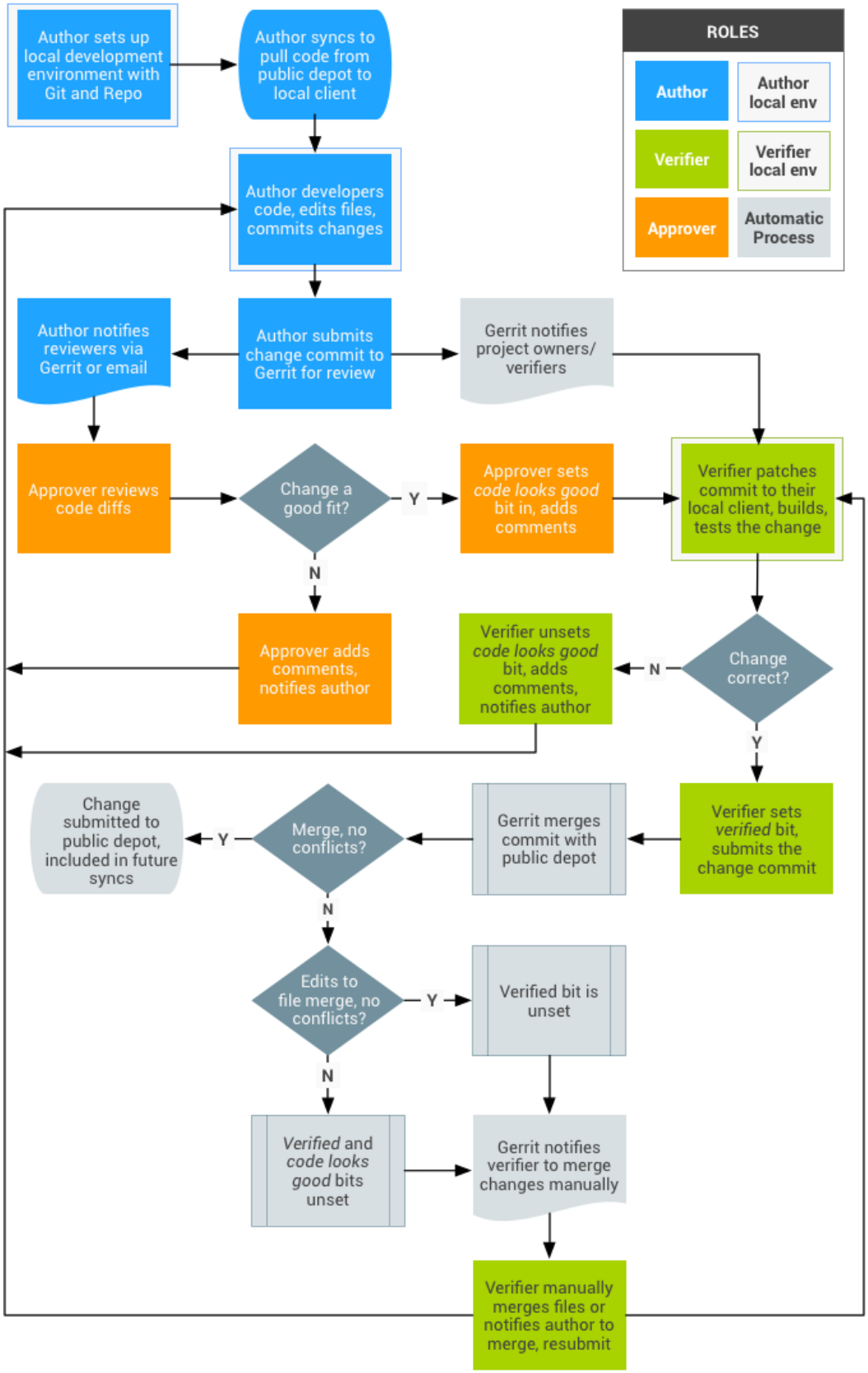


Figure 28 : Schéma du workflow d'un patch sur Gerrit selon Android[14]

Annexe 6 : Capture d'écran des Fragments de l'application finale

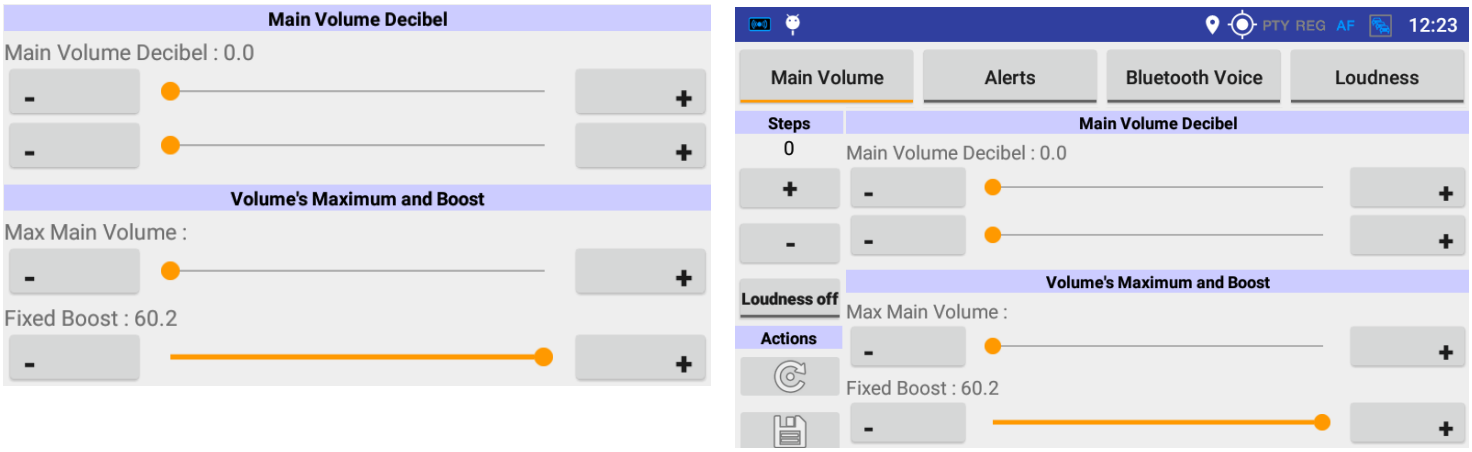
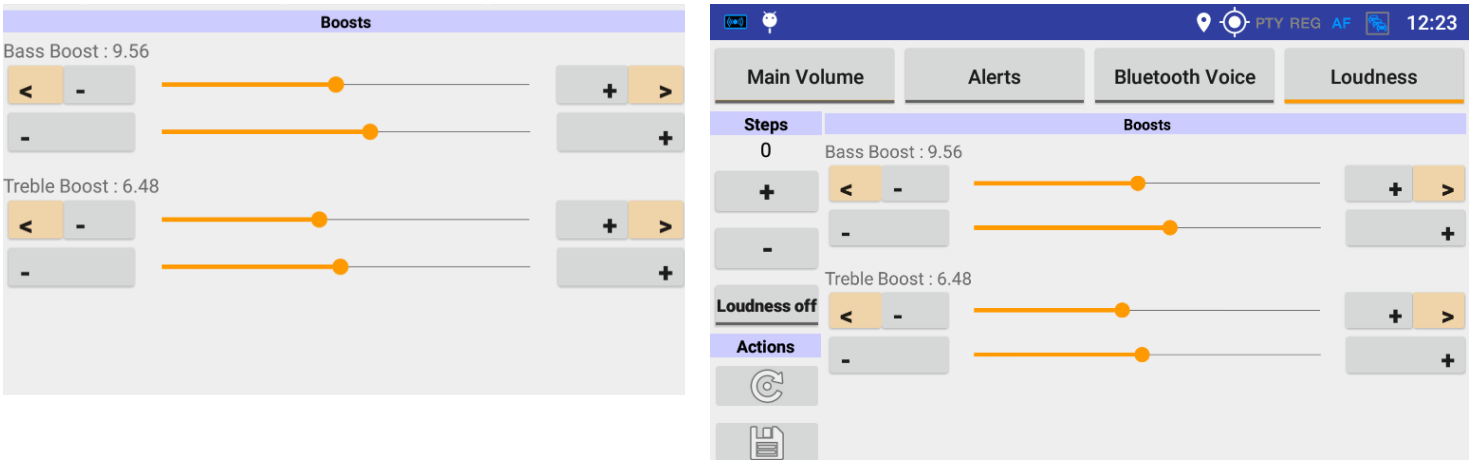


Figure 29 : Capture d'écran du layout généré depuis VolumesSettings.xml et son intégration dans l'application

Figure 30 : Capture d'écran du layout généré depuis LoudnessSettings.xml et son intégration dans l'application



Webographie

[1] http://insait.in/AIPA2012/articles/054.pdf -> Partie 4 : APPLICATIONS OF IOT

[2] http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing

[3] http://www.windriver.com/company/Wind-River-Corporate-Profile.pdf

[4] http://www.windriver.com/products/operating-systems/rocket/

[5] http://humanresources.about.com/od/glossaryf/g/flex_schedule.htm

[6] http://humanresources.about.com/od/employeebenefits/f/flex_schedules.htm

[7] http://fr.slideshare.net/asertseminar/embedded-system-in-automobiles

[8] https://developer.android.com/guide/components/fundamentals.html

[9] https://source.android.com/source/code-style.html#java-language-rules

[10] https://www.kernel.org/doc/Documentation/CodingStyle

[11] http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Why-Gradle-

[12] https://gradle.org/maven_vs_gradle/

[13] http://www.windriver.com/products/helix/

[14] https://source.android.com/source/life-of-a-patch.html