1. Environnement de travail
2. Définition

Android est un système d’exploitation Open Source pour Smartphones, PDA et terminaux mobiles conçu par Android, une startup rachetée par Google, et annoncé officiellement le 15 novembre 2007. Afin de promouvoir ce système d’exploitation ouvert, Google a su fédérer autour de lui une trentaine de partenaires réunis au sein de l’Open Handset Alliance

1. Avantage de cet environnement

Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes. Les Smartphones, les tablettes, la présence ou l'absence de clavier ou de trackball, différents processeurs... On trouve même des fours à micro-ondes qui fonctionnent à l'aide d'Android !

Non seulement c'est une immense chance d'avoir autant d'opportunités, mais en plus Android est construit de manière à faciliter le développement et la distribution en fonction des composants en présence dans le terminal (si votre application nécessite d'utiliser le Bluetooth, seuls les terminaux équipés de Bluetooth pourront la voir sur le Play Store).

1. La place d’Android dans ce projet

Dans ce projet je dois créer une application sur une tablette tactile ayant comme système d’exploitation Android 4.0.4.

Je dois donc comprendre le mode de fonctionnement d’Android lors de la création d’application.

Pour commencer les applications sont en Java. J’utilise éclipse, un éditeur open source, parce qu’il est adapté pour Android. Un écran virtuel est disponible pour pouvoir tester ces applications.

1. Le matériel utilisé

J’utilise une tablette tactile de marque Toshiba. Un écran de 10 pouces compose essentiellement la tablette.

1. Analyse
2. Les fonctions demandées lors de la 1ère itération

La fonction qui m’est demandées de la 1ère itération est: de faire réagir les projecteurs.

La fonction réagir projecteur consiste tous simplement lors d’une action sur la tablette envoyer les informations au serveur pour faire réagir les projecteurs, c’est-à-dire de les allumer, varier leurs intensités ou les éteindre.

1. Analyse UML
2. Gantt partie Android

Le diagramme prévisionnel est créé au début du projet et le planning réel au cours du développement de l’application.

On peut constater que les tests unitaires et d’intégration ont été longs. Effectivement j’ai dû passer plus de temps sur les tests unitaire que sur le développement de l’application.

Je sais, qu’à partir de la deuxième itération il va falloir que j’y consacre plus de temps.

|  |  |  |
| --- | --- | --- |
| Jour | **Planning prévisionnel Jimmy** | **Planning réel Jimmy** |
| Lundi 17/12 | Analyse du sujet | Analyse du sujet |
| Mardi 18/12 | Mise en place organisation | Mise en place organisation |
| Mercredi 19/12 | Analyse UML | Analyse UML |
|  |  |  |
| Lundi 07/01 | Analyse UML | Realisation Maquette IHM |
| Mardi 08/01 | Realisation Maquette IHM | Realisation Maquette IHM |
| Mercredi 09/01 | Realisation Maquette IHM | Realisation Maquette IHM |
|  |  |  |
| Lundi 14/01 | Realisation maquette IHM | Realisation de la connection socket |
| Mardi 15/01 | Realisation de la connection socket | Realisation de la connection socket |
| Mercredi 16/01 | Realisation de la connection socket | teste unitaire et d'intégration connection socket |
|  |  |  |
| Lundi 21/01 | Realisation de la connection socket | teste unitaire et d'intégration connection socket |
| Mardi 22/01 | Realisation de la connection socket | teste unitaire et d'intégration connection socket |
| Mercredi 23/01 | Ecriture du rapport et powerPoint | teste unitaire et d'intégration connection socket |
|  |  |  |
| Lundi 28/01 | Ecriture du rapport et powerPoint | teste unitaire et d'intégration connection socket |
| Mardi 29/01 | Ecriture du rapport et powerPoint | Ecriture du rapport |
| Mercredi 30/01 | Ecriture du rapport et powerPoint | Ecriture du rapport |
|  |  |  |
| Lundi 04/02 | Ecriture du rapport et powerPoint |  |

1. Diagramme des cas d’utilisations

Le diagramme des cas d’utilisations est composé de deux acteur et d’une action.

L’utilisateur tablette fait réagir les projecteurs et agis directement sur l’acteur projecteurs.



1. Diagramme de classe

J’ai réalisé au début de mon projet ce diagramme de classe pour l’application Android.

Cependant au cours de mon projet j’ai dû adapter le diagramme de classe à mes besoins et à l’architecture JAVA Android.

Au début du projet j’ai don pensé à un diagramme de classe avec 4 classes.

Une classe groupe (pour la deuxième itération) elle permet de rassembler les projecteurs en un groupe t il peut y avoir plusieurs groupe. Une classe projecteur et une classe socket pour faire réagir un projecteur (itération 1).

La classe Ihm représente le programme principale.



Maintenant mon diagramme de classe est adapté pour l’itération 1 Il comporte 4 classes aussi mais la classe groupe a été supprimer pour le moment. En revanche une classe pageAccueil a été ajoutée.

La classe projecteur a été remplacée par la classe seekbar elle représente l’élément seekbar avec essentiellement la progression. En Android 2.3.3 les Seekbars vertical ne sont pas implémentées par défaut.

La classe pageAccueil est une classe avec une IHM. Elle est correspond à la connexion de la tablette au serveur.

La classe socket a été complété avec un tableau de Bytes pour l’envoie des valeurs des projecteur, qui se font en tableau de Byte.

La classe IHM ne change pas de structure.



On peut donc constaté que l’analyse UML évolue lors de la création d’un projet.

1. Diagramme d’activité

Ce diagramme explique le déroulement de l’application.

Lors de l’exécution de l’application une connexion est obligatoire pour pouvoir utilisé les fonctionnalités de l’application.

Si une erreur survient lors de la connexion, l’application ne démarre pas et tente a nouveau de se connecter.

Après la connexion, on accède à la gestion de la lumière et on peut faire réagir un projecteur.

A la fin de l’action on peut recommencer à faire la gestion de la lumière.



1. Les scénarii

Scénarios nominaux

|  |  |  |  |
| --- | --- | --- | --- |
| Numéro action | Sujet | Verbe | Complément |
| 1  2 | Le client  Le client | Demare  Fait réagir | L’application  Le ou les projecteur(s) |

Scénarios d’exceptions

|  |  |  |  |
| --- | --- | --- | --- |
| Numéro action | Sujet | Verbe | Complément |
| 1  2 | Le client  Le client | N’a pas  n’arrive pas | Activé le wifi  A faire réagir le ou les projecteur(s) |

Scénarios alternatifs

|  |  |
| --- | --- |
| Numéro action | Action |
| 1  2 | L’application demande à l’utilisateur d’activé le wifi et propose un bouton pour accéder au paramètre  L’application se déconnecte et revient sur la page de connexion |

1. Application
2. Langage

Langage pour développer cette application est JAVA. C’est un langage POO (Programmation Orienté Objet).

Il est architecturé d’une façon précise :

Src : Ce dossier contient les fichiers de classe d’un programme.

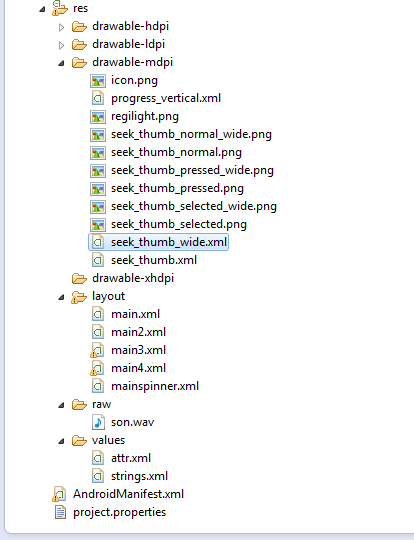
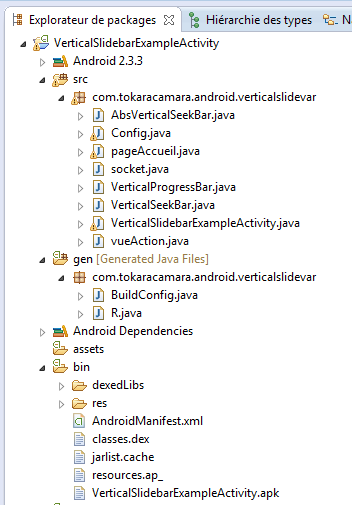
Gen : Ce dossier contient deux fichiers, BuildConfig et R, ils sont générés automatiquement par l’éditeur de texte. Ces fichiers correspondent aux adresses des objets du programme.

Assets : Ce dossier contient les fichiers multimédia d’un projet codé en 4.0 ou supérieur. Dans notre cas ce dossier n’est pas utile car je code en 2.3.3 et les fichiers multimédias ce trouve dans le dossier Res.

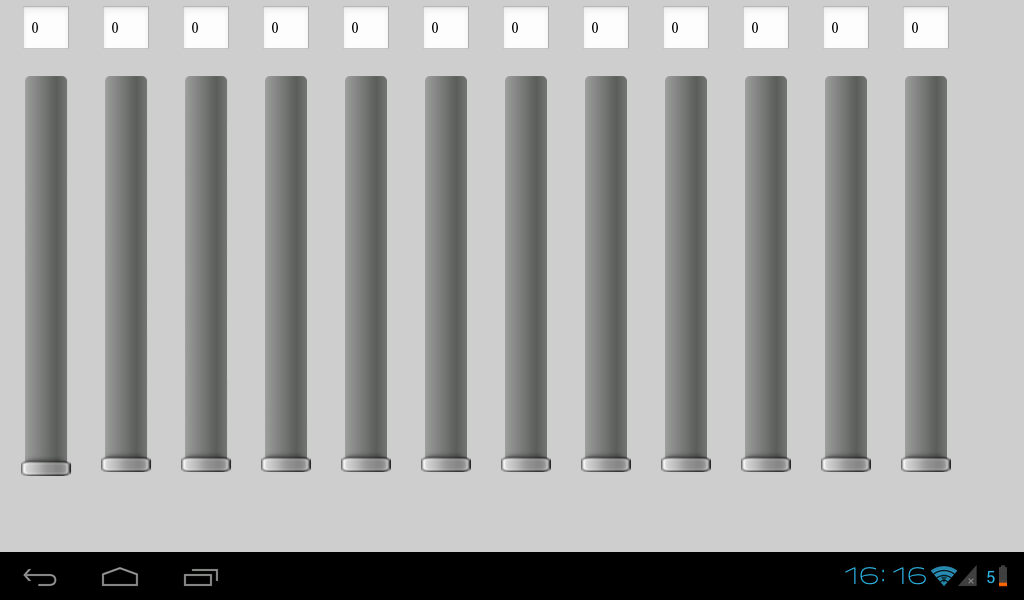
Bin : Ce dossier contient le fichier qui résulte d’une compilation vers l’exécutable Android.

Res : Ce dossier contient le fichier de l’interface graphique. Il a, lui-même, plusieurs sous dossiers comme drawable qui contient des images et qui en fonction du matériel choisi la résolution (Ldpi,mdpi,hdpi,xdpi), un dossier layout avec les fichiers de configuration de Template, un dossier raw pour le contenu multimédia et enfin le sous-dossier value qui contient toutes les définitions de variables

Et dans le dossier racine deux fichiers, le fichier manifeste indiquant le comportement de l’application (version, display, connectivité, stockage).



1. Interface graphique



Cette interface graphique ce compose de 12 seekbars (barre de contrôle de volume positionnée à la vertical).

Il a de zone de texte au dessue de chaque seekbar pour indiquer la puissance des projecteurs en pourcentage.

Lors du lancement de l’application une page de connexion s’ouvre et passe à la page de contrôle lorsque la connexion avec le serveur a été effectuée.

Si un problème réseau survient lors de l’utilisation de l’application, l’écran revient sur la page de connexion en attendant que le réseau revienne.

Des messages d’erreurs sont visibles lorsque la connexion est perturbée ou lorsque l’on quitte l’application. Ils sont de la forme d’une boite d’alerte.

1. Librairie

Les librairies utilisées sont :

* Android.app -> librairie graphique (alertDialogue, Activité)
* Android.view -> librairie graphique (positionnement)
* Android.widget -> librairie graphique (élément graphique)
* Android.util -> librairie debug (test, message console)
* Android.media -> librairie multimédia (jouer un son)
* Java.net -> librairie Java de réseau (socket)
* Java.io -> librairie d’exception (gestion erreur)

1. Différentes pages

Dans cette application il y a plusieurs vues, pages qui permettent à l’utilisateur de savoir sa situation à un moment t (connexion, écran de contrôle).

La première vue est la vue de connexion cette vue ne disparait pas tant que l’application n’est pas connecter au serveur. Car cette application a besoin de communiqué pour être fonctionnel.

Cette vue affiche un message d’erreur si l’utilisateur n’active pas le wifi.

La deuxième vue est la vue de contrôle de projecteur. Elle communique avec le serveur en envoyant les valeurs des seekbars dès que celle-ci sont actionnée. Une zone de texte affiche la valeur en pourcentage de chacun des projecteurs. Si l’utilisateur décide de quitter l’application un message de confirmation apparait.

Si la connexion avec le serveur est perdu, l’écran de contrôle se ferme et l’écran de connexion prend le relai et essaye de se connecter toute les 2 secondes s’il y a échec de la connexion.