[2014][TD8]

Magic tactil -

Magic Tactil est un jeu sur tablette et PC qui permet de jouer aux jeux de carte à collectionner. En plus, d'échanger des cartes et de s'amuser avec ses amis, le joueur pourra participer à la vie d'une communauté internationale. Pour cela, l'utilisateur discutera sur nos forums et pourra organiser des évènements.

21/03/2013

INTRODUCTION

Le projet consiste à réaliser une plate-forme basée sur les jeu de cartes. Cette dernière permettrait aux utilisateurs d'avoir les mêmes interactions que le vrai jeu de carte. L'application sera disponible sur Surface, Surface 2, Windows 8 ainsi que sur Android. Ce document a donc pour but d'apporter des explications sur cette plateforme et ainsi de guider l'utilisateur quand a son utilisation.

Description du document

Titre	[2014][TD8] Document Technique
Date	21/03/2013
Auteur	Oualid JOUHRI/Mehdi FARSI
Email	magictactil@epitech.eu
Version	6.0

Tableau des révisions

Date	Auteur	Section(s)	Commentaire
21/03/2013	Jouhri_o/farsi_m	Toutes	Première révision
20/05/2013	Periph_a	Windows	Ajout de ce qui concerne le client Windows
27/05/2013	Ortis_l	IOS	Ajout de la partie IOS
29/05/2013	Pucheu_m	Serveur	Ajout de la partie serveur
30/05/2013	Labori_b	Android	Ajout de la partie android
18/07/2013	Periph_a	Windows	Rooms
24/11/2013	Periph_a	Windows	Game + deck

			building
26/11/2013	Labori_b	Android	Ajout Game
13/12/2013	Pucheu_m	Serveur	Modification du module notification & libraire de test
01/01/2014	Labori_b	Android	Modification Module de jeu & librairie de test
07/01/2014	Ortis_l	IOS	Modification Module de jeu

Sommaire

Introduction

Magic Tactil est une application Client / Server, 3 clients sont en développement :

- Client Windows
- Client Android

- Client IOS

Serveur

Au lancement du serveur, un Thread sera lancé pour gérer la « Room » principale. « Room » qui contiendra tous les utilisateurs qui se connecteront.

Le but du Thread principal est d'accepter les différentes connections qui se font sur le serveur. Et dans ce même Thread, pour chaque client un nouveau Thread sera alloué pour ce même client.

Le module réseau

Le module réseau permet la lecture de Paquet. Lorsque Paquet est lu, il sera ainsi interprété puis ainsi donné au bon module qui l'utilisera.

Pour chaque paquet reçu une information sera renvoyée. Cette information contiendra la "réponse" à la requête de l'utilisateur.

Le module de notification

Ce module permet une interaction dynamisée entre les joueurs.

Elle peut être d'ordre sociale, les joueurs seront avertis lorsqu'un autre joueur communique avec eux par l'intermédiaire de la messagerie interne, ou encore, lorsqu'un joueur rejoint ou quitte une « room ».

Ces notifications sont également utilisées pendant une partie pour d'une part rendre le jeu plus fluide et d'autre part pour gérer des mécanismes simples comme :

- Bouger une carte d'une zone de jeu à une autre (la main, le deck, le cimetière ou l'exil)
- Engager une carte (tourner la carte pour qu'elle soit à l'horizontale)
- Dégager une carte (tourner la carte dans le sens contraire)
- Dégager toutes les cartes
- Mettre à jour des informations qui sont liées au jeu comme des points de vies
- Dessiner une flèche partant d'une carte et pointant vers une autre
- Abandonner une partie, annoncer à son adversaire que vous avez perdu la partie
- Proposer à son adversaire de recommencer une partie

Pour faire fonctionner les notifications, il est obligatoire que le paquet contiennent certaines informations.

Pour les notifications du jeu il est nécessaire de connaître l'expéditeur de la notification ainsi que le nom de la « room » dans laquelle il se trouve.

A cela des informations diverses vont entrer en jeu mais qui ne sont pas traiter par le serveur, comme les positions X et Y en pourcentage pour connaitre l'emplacement d'une carte ainsi que l'id qui lui est associé.

En ce qui concerne la messagerie, seuls l'expéditeur, le destinataire et le contenu du message sont nécessaire pour pouvoir utiliser le service.

Le module d'interprétation

Ce module permet de réorganiser les données entrantes et sortantes selon une norme déjà établie (Cf. La structure Paquet).

Dans le cas d'une donnée entrante, c'est le module réseau qui fera appel à ce service, par contre, si elle est sortante, ce sont les modules suivants qui

lui feront appel.

- Fonctions

Ces modules sont liés au module d'interprétation et à la base de données. Chaque module fera une requête à la base de données et en récupérera sa réponse.

Le module d'identification

Avant de pouvoir se connecter l'utilisateur doit créer un compte Magic Tactil, pour cela il faut les informations suivantes : nom, prénom, mail et mot de passe). Il est tout de même possible d'envoyer les autres informations dans le paquet (âge, sexe, lieu d'habitation).

Pour se connecter, le paquet doit contenir les informations suivantes : pseudonyme ou adresse mail et mot de passe.

Si les informations sont erronées, un code d'erreur sera renvoyé.

Le module de profile

Ce module permet de récupérer les informations concernant un utilisateur.

Pour cela, la donnée attendue est : pseudonyme de l'émetteur et pseudonyme de

l'utilisateur.

Si les données sont différentes, les informations concernant l'utilisateur seront renvoyées en fonction de ce qu'il a choisi de publier.

S'ils sont identiques toutes les informations seront renvoyées à l'exception du mot de passe.

En plus de cela, l'utilisateur a également la possibilité de changer ses informations personnelles.

Si le pseudonyme de l'utilisateur n'existe pas, un code d'erreur sera renvoyé.

Le module de "room"

Comme il a été précisé auparavant, lorsqu'un joueur se connecte, il sera automatiquement place dans une "room" principale.

Depuis cette "room" il est possible de :

- Créer une "room" avec : pseudonyme, nom de la "room"

- Joindre une "room" avec : pseudonyme, nom de la "room"

Depuis une nouvelle "room" il est possible de :

- Quitter une "room" avec : pseudonyme, nom de la "room"

- Détruire une "room" avec : pseudonyme, nom de la "room" et doit être le créateur de la "room" (visible avec la base de données)

Si une nouvelle "room" est détruite ou que l'utilisateur la quitte, il sera automatiquement réinséré dans la "room" principale.

Le module de discussion

Il permet l'envoie de message. Il y a plusieurs types d'envois qui sont :

- Message privé : nom de l'émetteur, nom du destinataire et le message

- Message dans une "room" : nom de l'émetteur, nom de la "room", message

Si la personne n'est plus dans la "room" ou que le destinataire n'est plus connecté, une erreur sera envoyée.

Le module de cartes

Ce module permet la recherche de cartes, les informations dont le module à besoin sont les caractéristiques de la carte recherchée, les champs possibles qui sont : nom, couleur, cout de mana, type, points (attaque et défense), texte, loyauté.

Si le module ne trouve pas la carte, un code d'erreur sera renvoyé.

Le module de Decks

Ce module permet la gestion totale des decks. Il est important de savoir que lorsque le joueur gagne une carte, elle sera automatiquement insérée dans un "Deck général".

Il est également possible, de créer différents types de decks, qui sont les decks "réels" qui seront basés sur les cartes acquises par le joueur et le deck de "vœux" qui sont seront basés sur les cartes que le joueur veut obtenir.

A chaque fois qu'une carte sera ajoutée dans un nouveau deck, elle sera automatiquement retirée du "Deck général".

Dans le cas contraire, elle sera automatiquement remise dans le "Deck général"

Il est possible d'effectuer diverses actions sur le "Deck General" et surtout les cartes qu'il contient qui sont :

- Créer un deck : pseudonyme utilisateur, nom du deck, bool

- Ajouter une carte au deck : pseudonyme utilisateur, nom du deck, id de la carte, nombre.

- Retirer une carte du deck : pseudonyme utilisateur, nom du deck, id de la carte, nombre.

Le module magasin

Ce module permet l'échange de cartes ou paquets de cartes entre les joueurs et Magic Tactil.

Magic Tactil propose trois types de ventes :

- La première étant la vente de carte à l'unité : pseudonyme de l'acheteur, id de la carte

- La seconde est la vente de cartes en paquet (15 cartes) : pseudonyme de l'acheteur, nom de l'édition et type d'achat ("paquet").

Pour ce type de vente, le serveur générera 15 cartes de façon aléatoire qui respecteront une norme à savoir que le paquet doit obligatoirement contenir au moins 1 carte rare, 3 cartes non communes et 12 communes. Il est possible d'obtenir 1 carte mythique, à ce moment-là, il n'y a pas de carte rare.

La troisième est la vente de boites (36 paquets) : pseudonyme de l'acheteur, nom de l'édition, type d'achat ("boite").

Même principe que pour le paquet sauf que ce procédé sera répété 36 fois. Tout comme le paquet, la boite doit respecter une norme, en effet, la boite doit obligatoirement être constituée de 4 cartes rares mythiques.

Il est tout de même possible pour les joueurs de vendre et d'acheter des cartes à d'autres utilisateurs.

Pour cela, Magic Tactil propose un système d'enchère. Ce système pourra supporter 2 types d'enchères :

- Enchère classique ("EC")

- Achat direct ("AD")

L'utilisateur pourra donc :

- Mettre en enchère une carte : pseudonyme, nom de la carte, édition, temps, prix, type d'enchères.

- Enchérir sur une carte : pseudonyme de l'émetteur, id de l'enchère, prix

- Le Paquet

Le Paquet est une structure qui nous sert de norme de communication.

La structure Paquet est constitué de :

- Source

- Destination

- Code de fonctions en 4 lettres

- Données

- La donnée du Paquet

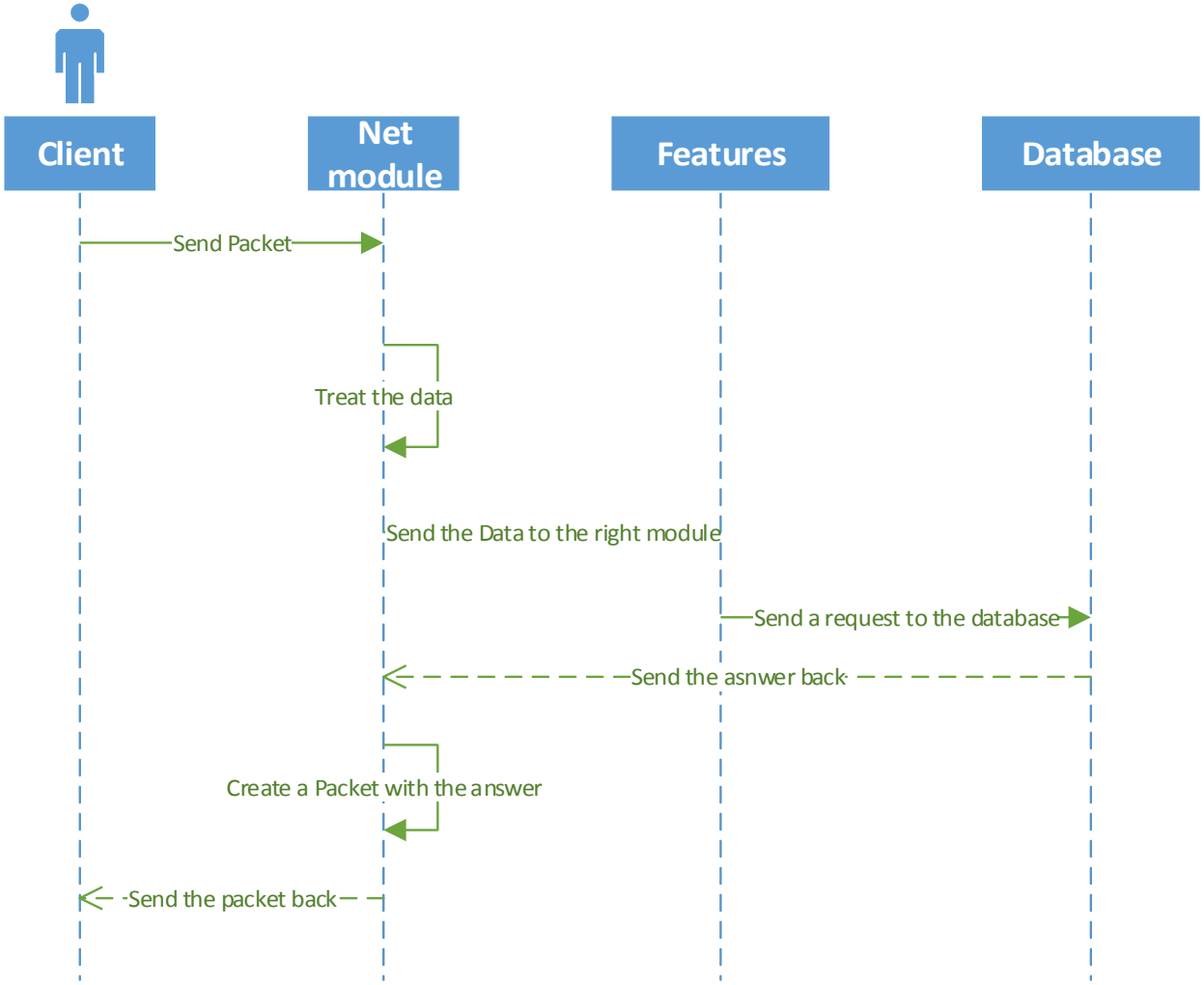
La donnée du Paquet doit être construite de façon spéciale qui est :

- clef\rvaleur\nclef2\rvaleur\n

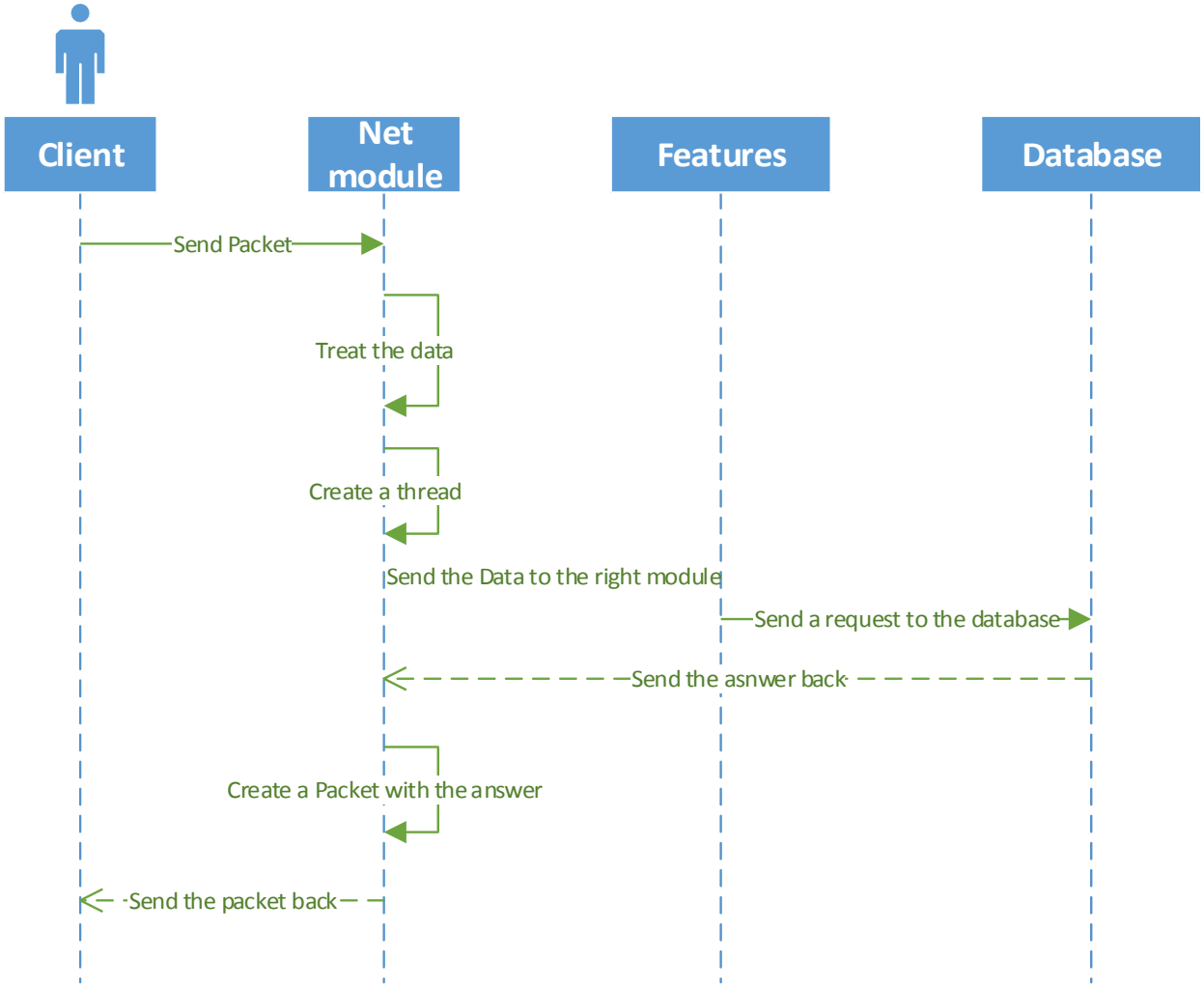
Si une erreur se produit la donnée vaudra ("KO").

Comment ça marche ?

Le diagramme suivant montre les différentes étapes de fonctionnement lorsque le serveur reçoit une requête d'un des utilisateurs.



Seulement dans le cas de la connexion d'un client à notre serveur, la réaction est un peu différente comme le montre le diagramme suivant.



Nous avons donc vu que dans le cas d'une connexion, le serveur créera un thread associé au client connecté.

Client Windows

Le client Windows est compatible avec Windows 7 et Windows 8. Il utilise le SDK Surface 2.

Un pattern MVVM (~ MVC), a été mis en place. La partie View est en XAML, le ViewModel et le Model en C#, le model gère en réalité la communication avec le serveur.

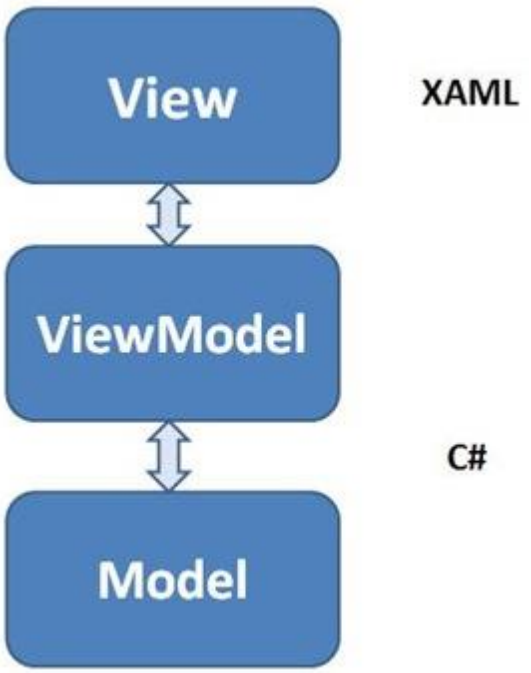
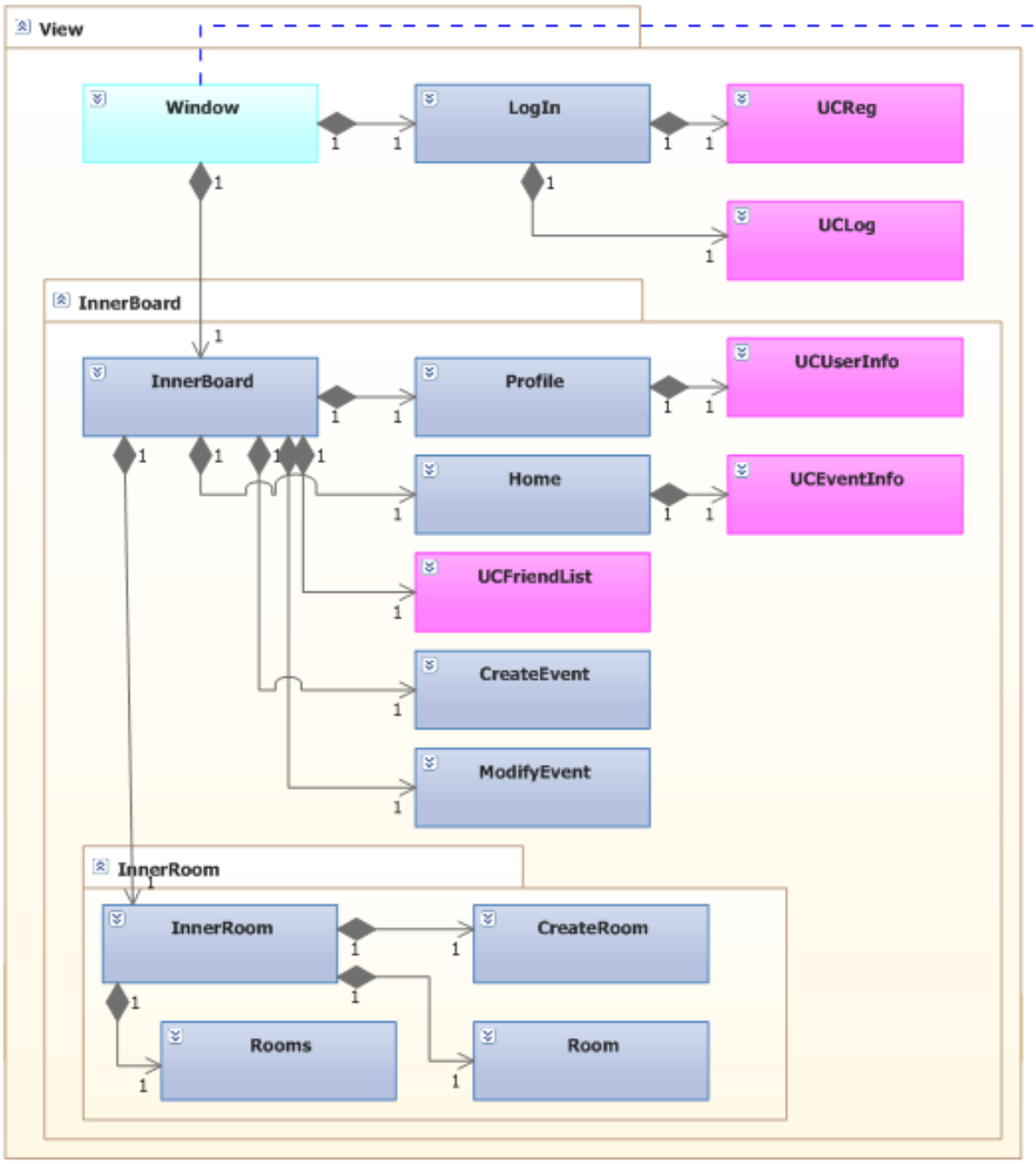


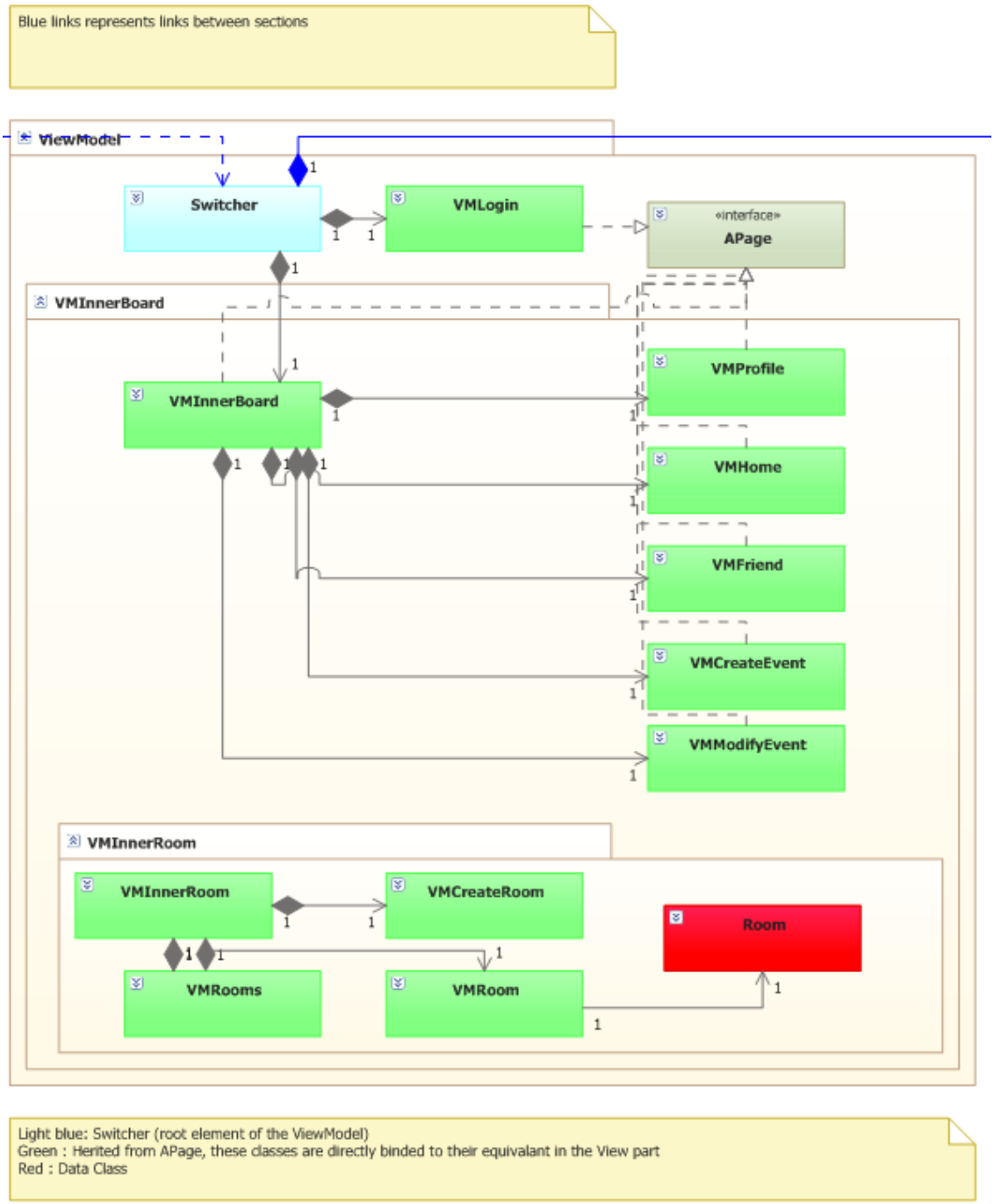
Illustration : Pattern MVVM

Organisation générale

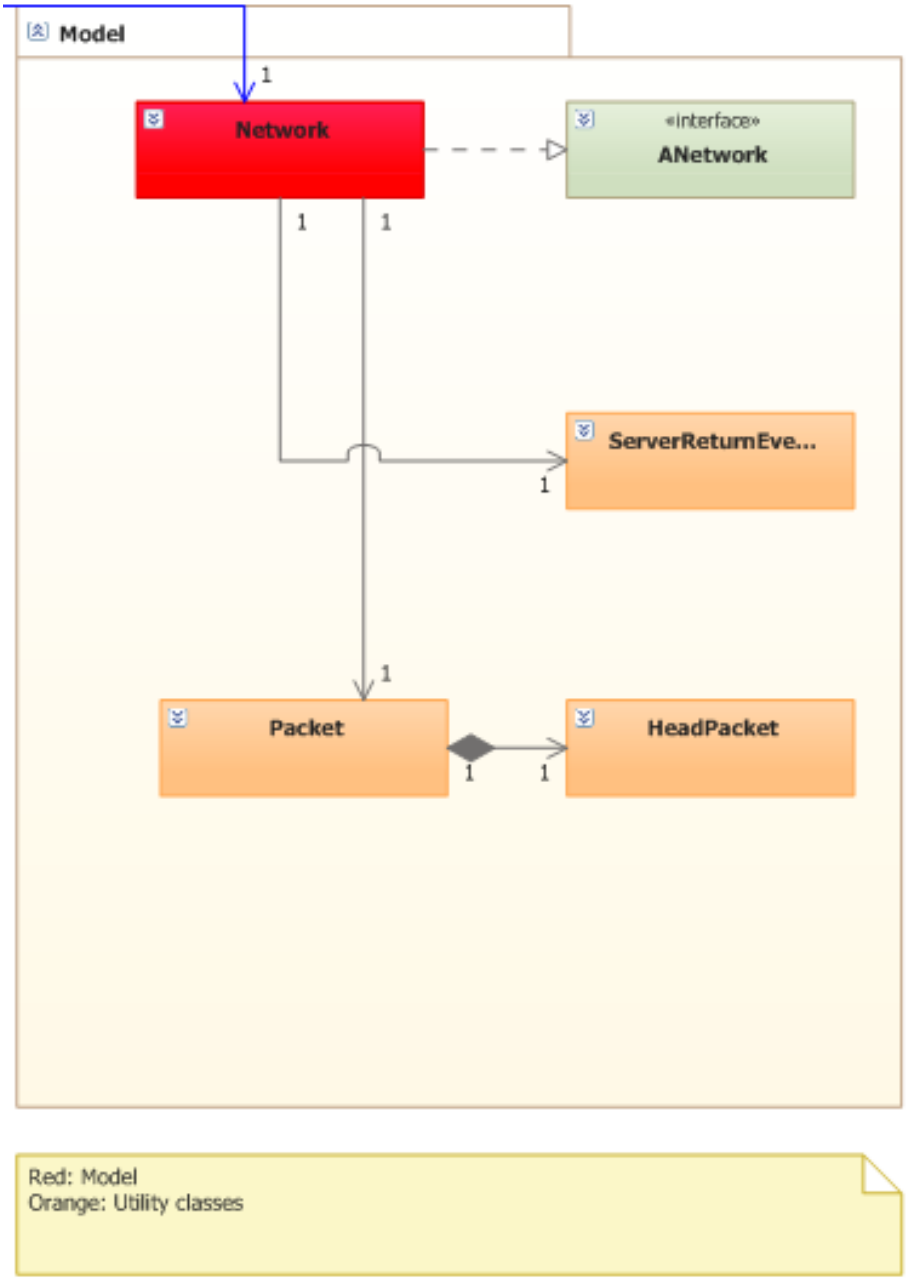
View


View
Window
LogIn
UCReg
UCLog
InnerBoard
InnerBoard
Profile
UCUserInfo
Home
UCEventInfo
UCFriendList
CreateEvent
ModifyEvent
InnerRoom
InnerRoom
CreateRoom
Rooms
Room

ViewModel



Model



Module Network

Le module Network est le module qui communique avec le serveur. Ce module et la partie ViewModel forment un Observer pattern. Ceci permet un retour asynchrone des requêtes faites au serveur.

Le module Network dispose d'un thread d'envoi vers le serveur et d'un thread de réception des messages du serveur.

Pour chaque requête possible vers le serveur, il y a dans le module Network une méthode et un Event associés. Quand le ViewModel appelle une méthode du module Network, le thread d'envoi fait la requête au serveur. Puis quand le thread de réception reçois la réponse du serveur, il throw l'évènement associé.

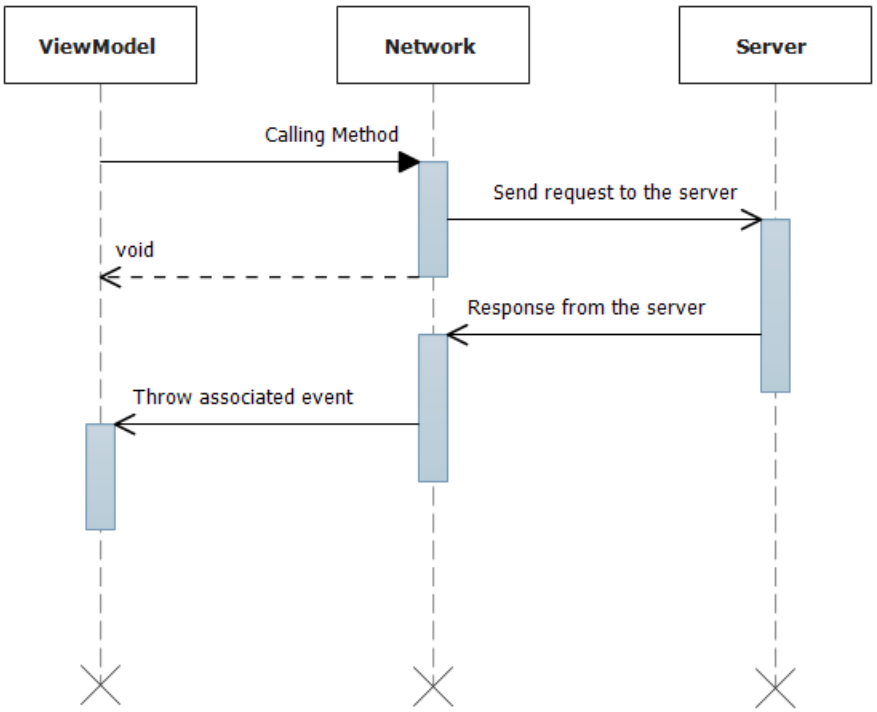


Illustration de l'Observer pattern mis en place.

Coté ViewModel, il faut que le module qui utilise la méthode de Network ce soit au préalable abonné à l'event associé.

Module Login

Ce module sert à se connecter au serveur, se déconnecter ou créer un compte. Le module Login est composé de la classe VMLogin(C#) dans le View Model et de l'interface utilisateur LogIn (xaml) dans le View. Les deux entités sont liées par des bindings.

VMlogin(C#)	LogIn(xaml)	Utilité ViewModel	Utilité View
Boolean __RBlogIn	RadioButton radioLogin	Appelle la méthode de login ou la méthode de création de compte en fonction de la valeur.	Affiche le formulaire de login ou de création de compte (register) en fonction de la valeur.
String name	Textbox name	Utilisé pour communiquer avec le model	Permet à l'utilisateur d'écrire son username.
String password	PasswordBox password	Utilisé pour communiquer avec le model	Permet à l'utilisateur d'écrire son password.
String tmppassword	PasswordBox confirmPassword	Utilisé pour vérifier qu'il n'y a pas d'erreur de password lors de la création de compte	Permet à l'utilisateur d'écrire une deuxième fois son password lors de la création de compte.
String email	TextBox email	Utilisé pour communiquer avec le model lors de la création de compte	Permet à l'utilisateur d'écrire son email lors de la création de compte.
Action submit	Button Submit	Apelle la méthode du Model	Permet à l'utilisateur de confirmer ses

			informations et de se connecter ou créer un compte

Module Profile

Ce module sert à éditer ses informations personnelles. Le module profile est composé de la classe VMProfile(C#) dans le View Model et de l'interface utilisateur Profile (xaml) dans le View. Les deux entités sont liées par des bindings.

La modification des valeurs sur l'interface entraine l'appel des fonctions du serveur.

VMProfile(C#)	Profile(xaml)
String username	Label/TextBox username
String mail	Label/TextBox mail
String firstName	Label/Textbox firstname
String surname	Label/TextBox surname
String birth	Label/TextBox birth
String location	Label/TextBox location
String phone	Label/Textbox phone
String gender	Label/Textbox gender
Action edit	Button edit

Avant d'être transmises au model les nouvelles entrées sont vérifiées pour correspondre aux standards du serveur.

Module Friend

Ce module sert à gérer une lite d'amis. Le module friend est composé de la classe VMFriend(C#) dans le View Model et du contrôle utilisateur UCFriendList (xaml) dans le View. Les deux entités sont liées par des bindings.

VMFriend(C#)	UCFriendList(xaml)	Utilité
Boolean frORbl	RadioButton frORbl	Permet d'afficher la liste d'amis ou la liste noire
ObservableCollection<String> frORblList	ListBox friendListBox	Contiens la liste noire ou la liste d'amis en fonction de la valeur de frORbl
String friend	TextBox friend	L'utilisateur peut rentrer le nom de l'ami à ajouter
Action addFriend	Button addFriend	Ajoute la valeur 'friend' à la liste d'amis ou à la liste noire en fonction de frORbl

Module Home

Ce module affiche la liste des events et leurs informations. Le module Home est composé de la classe VMHome(C#) dans le View Model et de l'interface utilisateur Home (xaml) dans le View. Les deux entités sont liées par des bindings.

VMHome(C#)	Home(xaml)	Utilité
ObservableList<String> events	ListBox events	La liste des noms des events sur le serveur.
String eventName	TextBox eventName	Le nom de l'event sélectionné.
String creatorName	TextBox creatorName	Le nom du créateur de l'event sélectionné.
String date	TextBox date	La date de l'event sélectionné
String location	TextBox location	Le lieu de l'event sélectionné
String content	TextBox content	La description de l'event sélectionné

Module CreateEvent

Ce module permet de créer un event. Le module CreateEvent est composé de la classe VMCreateEvent(C#) dans le View Model et de l'interface utilisateur CreateEvent (xaml) dans le View. Les deux entités sont liées par des bindings.

VMCreateEvent(C#)	CreateEvent(xaml)
String eventName	TextBox eventName
String date	TextBox date
String location	TextBox location
String content	TextBox content
Action Create	Button create
Action Cancel	Button cancel

Module ModifyEvent

Ce module permet de modifier un event dont on est le créateur. Le module ModifyEvent est composé de la classe VMModifyEvent(C#) dans le View Model et de l'interface utilisateur ModifyEvent (xaml) dans le View. Les deux entités sont liées par des bindings.

VMModifyEvent(C#)	ModifyEvent(xaml)
String date	TextBox date
String location	TextBox location
String content	TextBox content
Action Edit	Button edit
Action Cancel	Button cancel

Module CreateRoom

Ce module permet de créer une room. Le module CreateRoom est composé de la classe VMCreateRoom(c#) dans le View Model et de l'interface utilisateur CreateRoom(xaml) dans le View. Les deux entités sont liées par bindings.

VMCreateRoom(C#)	CreateRoom(xaml)
String roomName	TextBox roomName
String format	SurfaceComboBox format
Action Create	Button Create

Module Rooms

Ce module permet de consulter les rooms et de rejoindre une room. Le module Rooms est composé de la classe VMRooms(c#) dans le View Model et de l'interface utilisateur Rooms(xaml) dans le View. Les deux entités sont liées par bindings.

VMRooms(C#)	Rooms(xaml)
List<String> rooms	SurfaceListBox rooms
Action Join	Button Join

Module Room

Ce module permet de quitter sa room. Le module Room est composé de la classe VMRoom(c#) dans le View Model et de l'interface utilisateur Room(xaml) dans le View. Les deux entités sont liées par bindings.

VMRooms(C#)	Rooms(xaml)
Action leave	Button Leave

Create Deck module

Ce module permet de créer un deck. Le module createDEck est composé de la classe VMInnerDeck(c#) dans le View Model et de l'interface utilisateur Innerdeck(xaml) dans le View. Les deux entités sont liées par bindings.

VMInnerDeck(C#)	InnerDecks (xaml)
String deckname	Textbox name
List<Card> deck	SurfaceListBox deck
List<Card> side	SurfaceListBox side

Game Module

Ce module permet de jouer. Le module game est composé de la classe VMGame(c#) dans le View Model et de l'interface utilisateur Game(xaml) dans le View. Les deux entités sont liées par bindings.

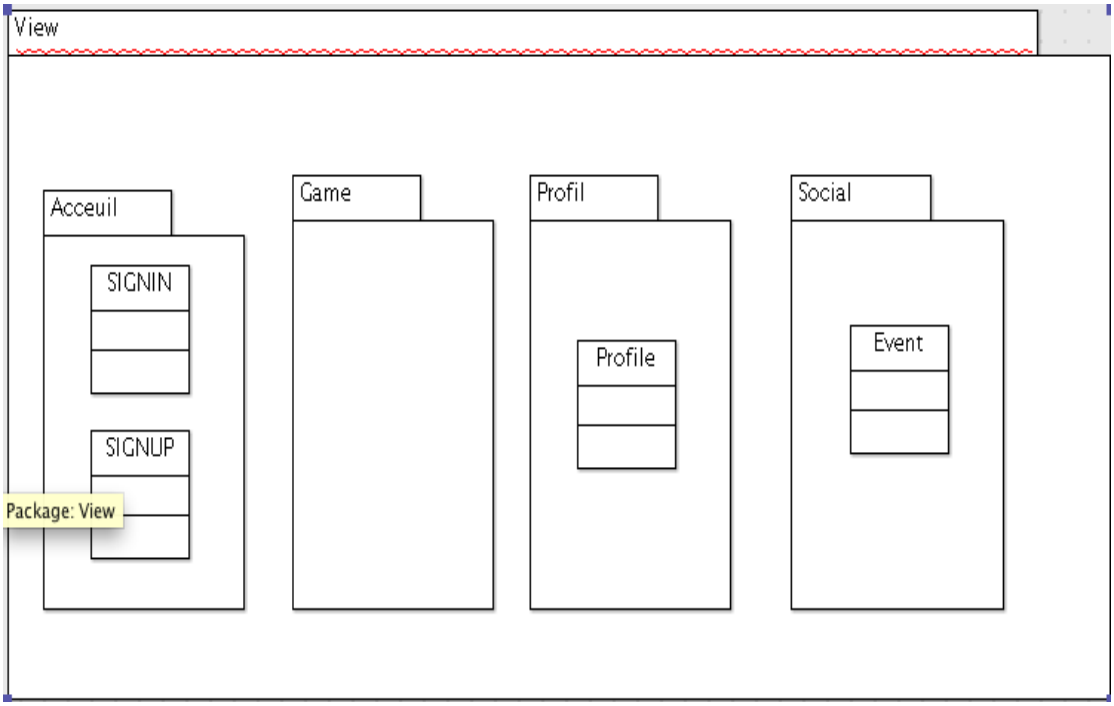
VMGame(C#)	Game (xaml)
List<card> hand	Surfacelistbox hand
List<Card> deck	SurfaceListBox deck
List<Card> field	SurfaceListBox field
List<card> graveyard	Surfacelistbox graveyard
List<card> exile	Surfacelistbox exile

Client IOS

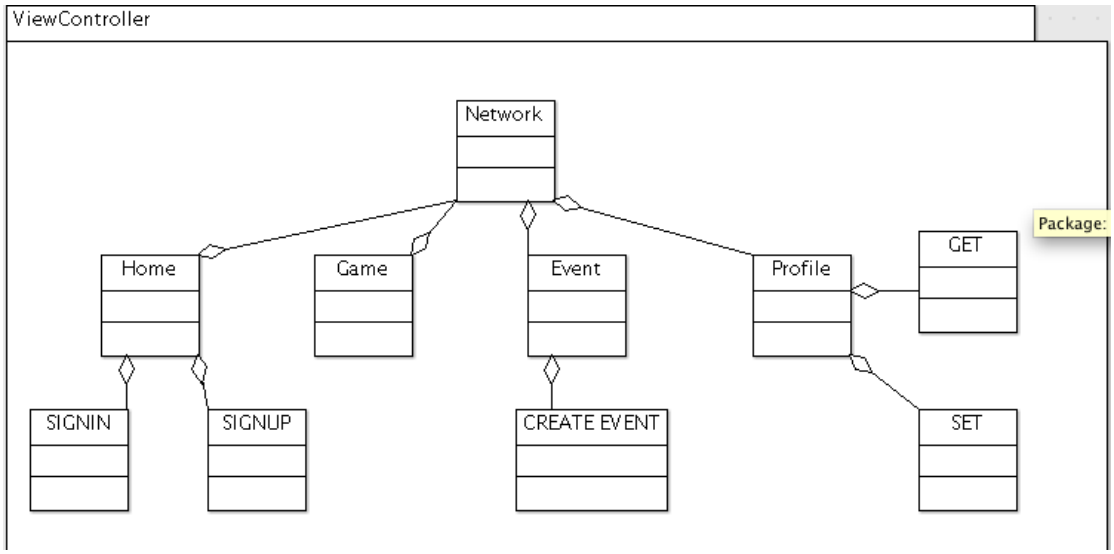
Le client IOS est compatible iPhone (3G, 3GS, 4, 4S, 5), iPad (1, 2, Rétina, Mini) et iPod Touch. Il est développé avec les technologies officielles d'Apple en Objective-C avec Xcode 4.6.2. Il faudra au moins disposer d'IOS 6.0 sur son iDevice pour exécuter l'application.

Un design pattern MVC (Model View Controller) a été implémenté. La view est généré en XML, le viewController ainsi que le model sont en Objective-C. Le viewController gère les évènements de la view (touch, multiple touch, scroll, etc…) et le model est le pont reliant l'application au serveur.

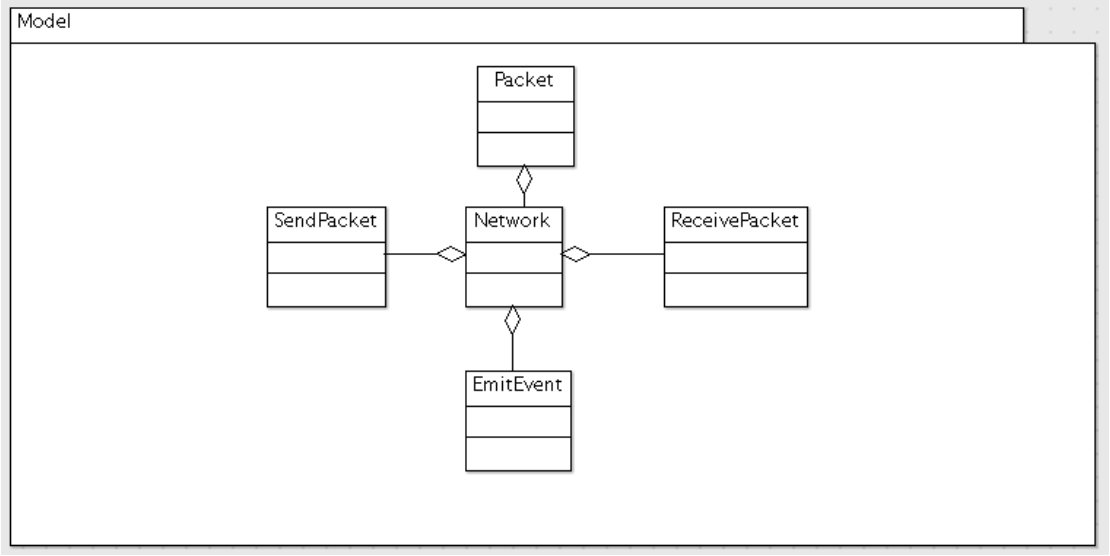
VIEW :



VIEW CONTROLLER :

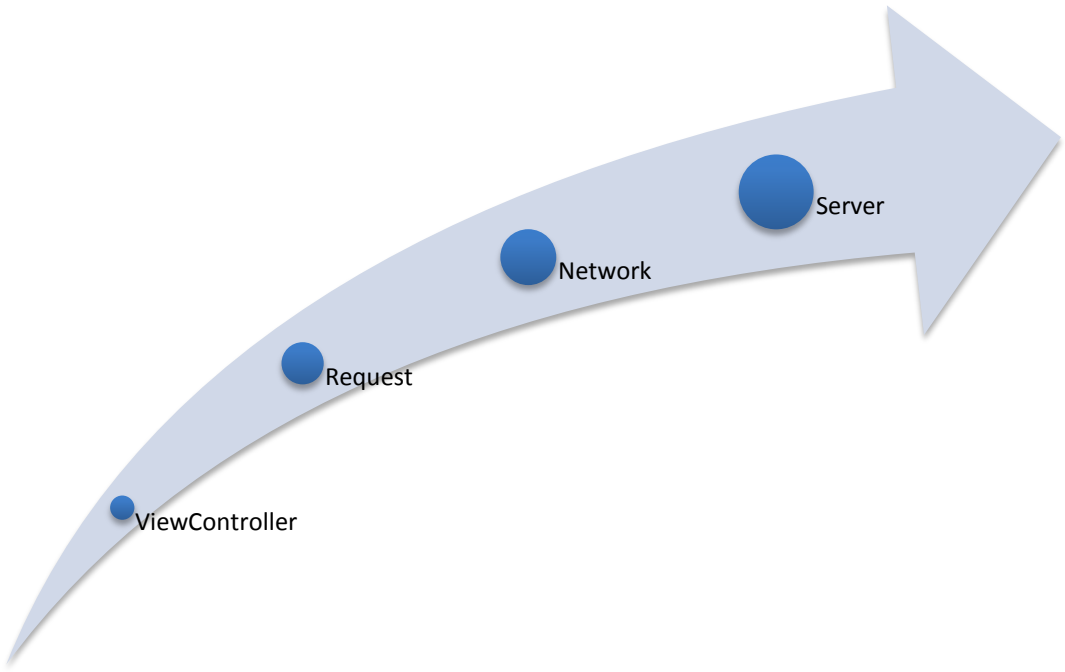


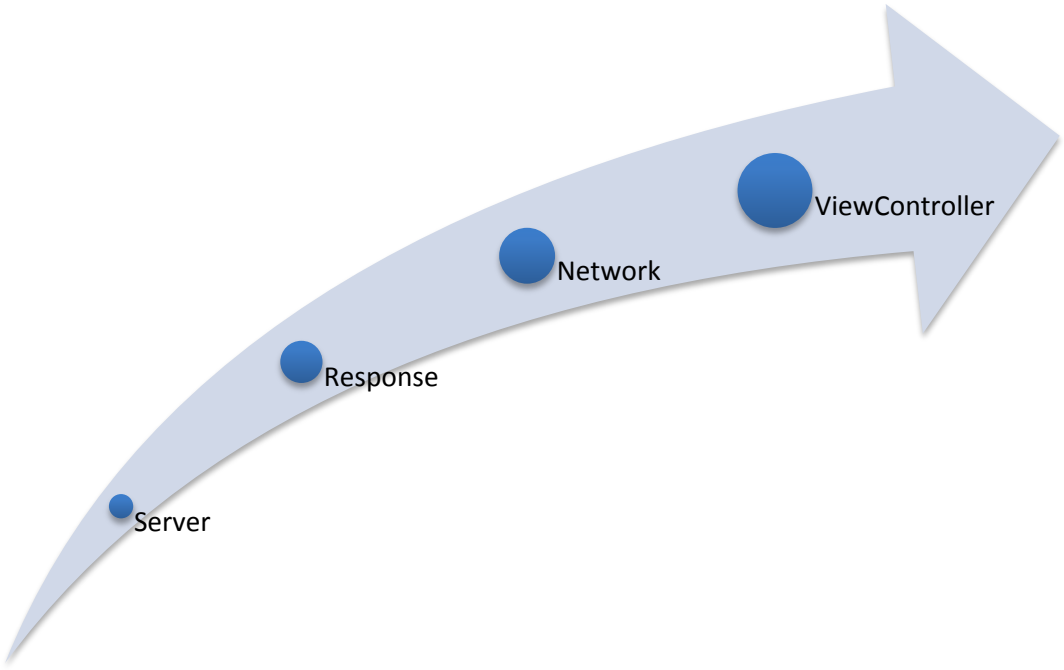
MODEL :



Module Network :

Le module Network permet de manière asynchrone d'envoyer et de recevoir des paquets au serveur. Il utilise une connexion TCP classique. Le module Network reçoit les paquets en temps réel et envoi les signaux correspondants afin de réveiller les modules correspondants. Il utilise un système de producteur-consommateur à l'aide d'une queue qui est approvisionné par le serveur et consommé par les modules correspondants.


Server
Network
Request
ViewController


ViewController
Network
Response
Server

Module Login :

Le module Login permet de s'inscrire, se connecter ou se déconnecter du serveur. Il est composé des classes SIGNIN et SIGNUP qui sont chacune associées à une vue.

VClogin(Objective C)	SIGNIN(xml)	Utilité ViewController	Utilité View
UITabBard	UIButton	Appelle la méthode de login ou la méthode de création de compte en fonction du button selectionné	Affiche le formulaire de login ou de création de compte.
NSString name	UITextField name	Utilisé pour communiquer avec le model	Permet à l'utilisateur d'écrire son username.
NSString password	UITextField password	Utilisé pour communiquer avec le model	Permet à l'utilisateur d'écrire son password.
NSString Confirmpassword	UITextField confirmPassword	Utilisé pour vérifier qu'il n'y a pas d'erreur de password lors de la création de compte	Permet à l'utilisateur d'écrire une deuxième fois son password lors de la création de compte.
NSString email	UITextField email	Utilisé pour communiquer avec le model lors de la création de compte	Permet à l'utilisateur d'écrire son email lors de la création de compte.
NSEvent submit	UIButton Submit	Apelle la méthode du Model	Permet à l'utilisateur de confirmer ses informations et de se connecter ou créer un compte

Module Profile :

Le module profile permet de voir toutes les informations du compte, il permet aussi d'éditer ses informations. Il est composé de la classe Profile qui est associé à une vue.

VCProfile(Objective C)	Profile(XML)
NSString username	UILabel/UITextField username
NSString mail	UILabel/UITextField mail
NSString firstName	UILabel/UITextfield firstname
NSString surname	UILabel/UITextField surname
NSString birth	UILabel/UITextField birth
NSString location	UILabel/UITextField location
NSString phone	UILabel/UITextField phone
NSString gender	UILabel/UITextField gender
NSEvent update	UIButton update

Module CreateEvent :

Ce module permet de créer un événement. Ce module est composé des Classes Events, EventObject, AddEvent qui sont chacune associés à une vue.

VCCreateEvent(Objective-C)	AddEvent(XML)
NSString eventName	UITextField eventName
NSString date	UITextField date
NSString location	UITextField location
NSString content	UITextEdit content
NSEvent Add	UIButton add
NSEvent Back	UIButton back

Module UpdateEvent

Ce module permet de modifier un event dont on est le créateur. Le module UpdateEvent est composé de la classe UpdateEventView dans le View Model et de l'interface utilisateur EditEventView dans le View. Les deux entités sont liées par des bindings.

UpdateEvent	UpdateEventView
NString date	TextField date
NString location	TextField location
NString content	TextField content
IBAction Edit	UIButton edit
IBAction Cancel	UIButton cancel

Module CreateRoom

Ce module permet de créer une room. Le module CreateRoom est composé de la classe CreateRoom dans le View Model et de l'interface utilisateur CreateRoomView dans le View. Les deux entités sont liées par bindings.

CreateRoom	CreateRoomView
NString roomName	TextField roomName
NString format	Segmented Control
IBAction Create	UIButton Create

Module Rooms

Ce module permet de consulter les rooms et de rejoindre une room. Le module Rooms est composé de la classe Room dans le View Model et de l'interface utilisateur RoomView dans le View. Les deux entités sont liées par bindings.

Room	RoomView
NSArray *rooms	TableView *room
IBAction Join	UIButton Join

Module JoinRoom

Ce module permet de quitter sa room. Le module Room est composé de la classe JoinRoom dans le View Model et de l'interface utilisateur JoinRoomView dans le View. Les deux entités sont liées par bindings.

JoinRoom	JoinRoomView
IBAction leave	UIButton Leave

Create Deck module

Ce module permet de créer un deck. Le module createDEck est composé de la classe CreateDeck dans le View Model et de l'interface utilisateur CreateDeckview dans le View. Les deux entités sont liées par bindings.

CreateDeck	CreateDeckView
NSString deckname	TextField name
NSArray deck	UICollection deck
Bool side	Segmented control side

Game Module

Ce module permet de jouer. Le module game est composé de la classe Game dans le View Model et de l'interface utilisateur GameView dans le View. Les deux

entités sont liées par bindings. Le module permet d'éditer son nombre de point et de définir un vainqueur.

Game	GameView
NSArray hand	UIScrollView hand
NSArray deck	UIScrollView deck
NSArray field	UIScrollView field
NSArray graveyard	UIScrollView graveyard
NSArray exile	UIScrollView exile
Int points	UILabel points

Les Tests

Toute une série de tests unitaires ont été implémentés, vérifiant la stabilité, la rapidité, et la fiabilité des différentes fonctionnalités de l'application. Les tests unitaires ont étés fait avec le plugin natif XCTest disponible depuis XCode 5.

Client Android

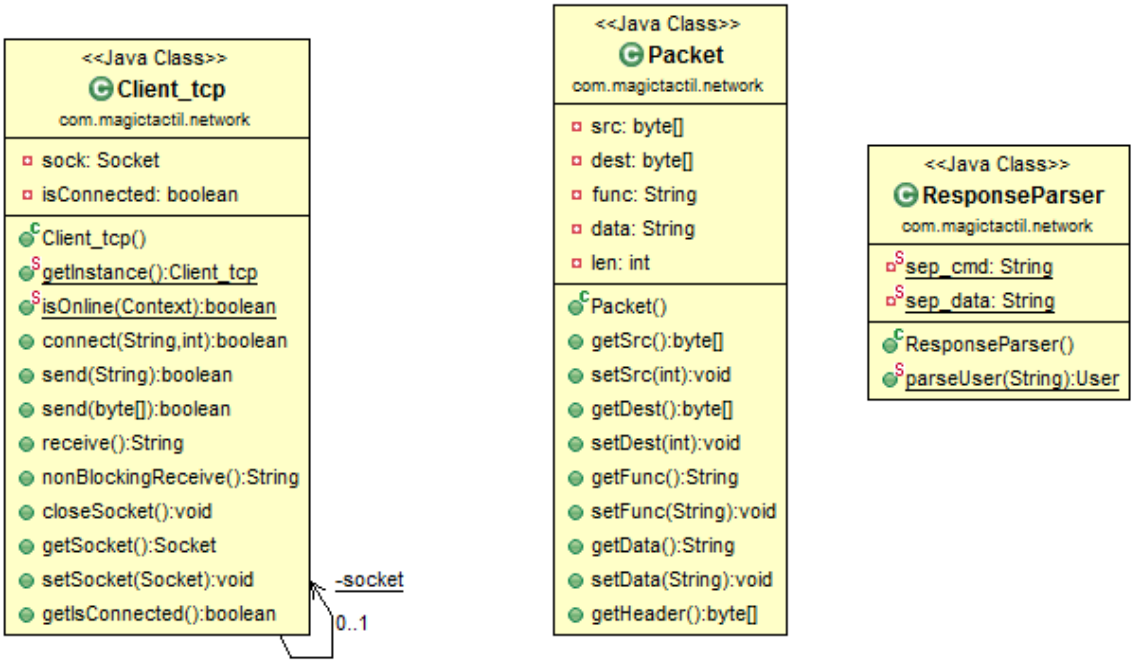
L'application Android MagicTactil est supporté à partir de la version 8 de l'API Android soit des téléphones à partir de la version 2.2.x du nom de FROYO. Elle est aussi adaptée aux versions tablette.

Le design pattern MVC (Model View Controller) est utilisé. Les vue (layouts) sont en XML. Les modelés et les contrôleurs sont en JAVA.

Les librairies supplémentaires utilisées pour l'application Android sont :

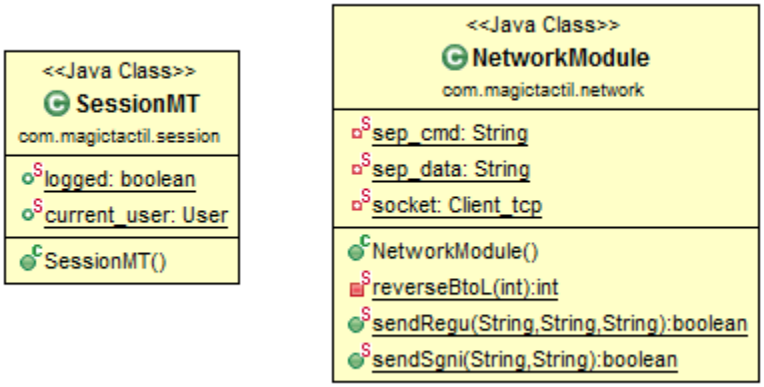
- ActionBarSherlock : permet d'utiliser l'action bar sur toutes les versions d'Android.
- SDK Facebook : Permet d'utiliser l'API Facebook
- ANDEngine : 2D OpenGl game Engine
- Robotium : tets de scénarios

Network



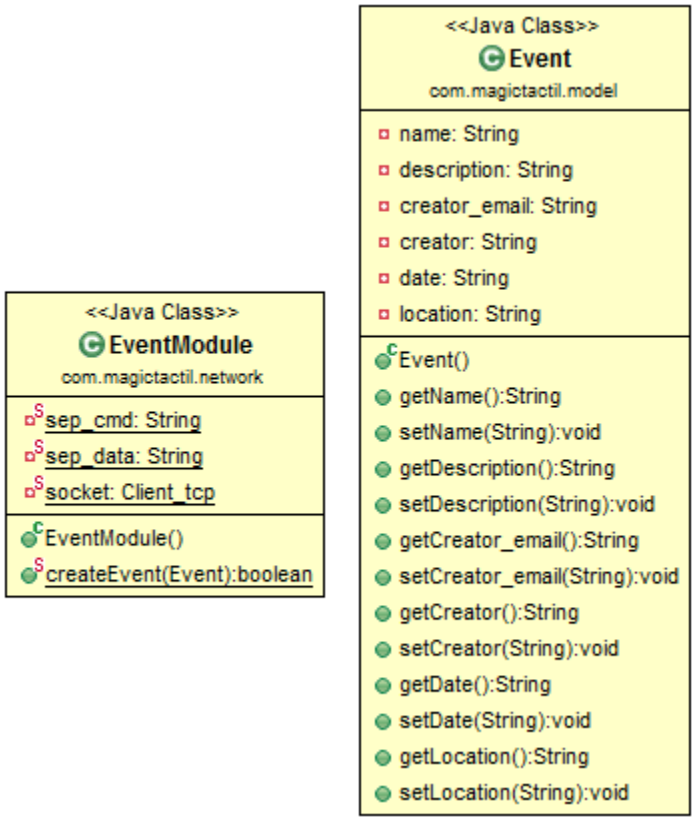
Module Login

Le module Login permet la connexion à son compte et la création de compte sur le serveur.



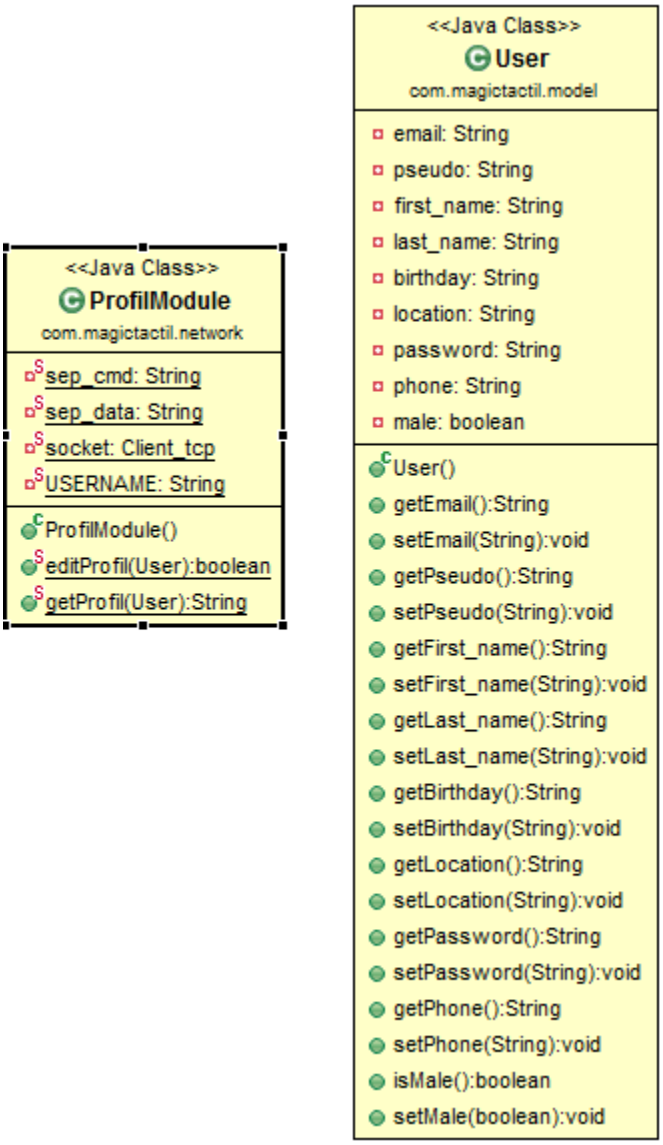
Module Event

Le module Event permet pour l'instant de créer des évènements.



Module Profile

Le module profile permet de voir toutes les informations du compte, il permet aussi d'éditer ses informations.



Module Game

La partie de jeu se base sur le moteur graphique AndEngine lui-même basé sur Open GL. Il permet une grande flexibilité et une plus grande réactivité côté jeu.

Le module jeu est composé de 2 classes Player pour les joueurs, cette classe contient une classe Hand pour son jeu, une classe Deck pour son deck de type LIFO (Last In First Out), points de vie, etc …

Les Tests

En plus des tests unitaires Junit le projet contient des tests dit de scénario. Ces tests sont mis en place avec la librairie Robotium.

L'application Android est aussi soumis au « Monkey « test.