|  |
| --- |
| [2014][TD5] |
| Magic tactil - |
| Magic Tactil is a game on tablet and PC for playing trading card game. The player would be able to trade cards, have fun with friends but also participate in the life of an international community. So the user would discuss on our forums and be able to organize events. |

21/03/2013

INTRODUCTION

The project consists of a platform based on the card games. This application would allow users to play the game as in real life. The application will be available on IOS, Surface, Pixel Sense, Windows 8 as well as Android. This paper is intended to provide technical explanations about this platform.

Document description

|  |  |
| --- | --- |
|  |  |
| Title | [2014][TD5] |
| Date | 21/03/2013 |
| Auteur | Oualid JOUHRI/Mehdi FARSI |
| Email | [magictactil@epitech.eu](mailto:magictactil@epitech.eu) |
| Version | 6.0 |

Revision Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Author | Section(s) | Comment |  |
| 21/03/2013 | Jouhri\_o/farsi\_m | All | First revision |  |
| 20/05/2013 | Periph\_a | Windows | Windows Client part |  |
| 27/05/2013 | Ortis\_l | IOS | IOS part |  |
| 29/05/2013 | Pucheu\_m | Server | Server part |  |
| 30/05/2013 | Labori\_b | Android | Android part |  |
| 18/07/2013 | Periph\_a | Windows | Rooms |  |
| 24/11/2013 | Periph\_a | Windows | Deck building + game |  |

Summary

[Introduction 6](#_Toc373067362)

[The server 6](#_Toc373067363)

[The network module 6](#_Toc373067364)

[The notification module 6](#_Toc373067365)

[The interpretation module 7](#_Toc373067366)

[Authentication module 7](#_Toc373067367)

[Profile module 8](#_Toc373067368)

[Room module 8](#_Toc373067369)

[Chat module 8](#_Toc373067370)

[Cards module 8](#_Toc373067371)

[Deck module 9](#_Toc373067372)

[Shop module 9](#_Toc373067373)

[How does it work? 10](#_Toc373067374)

[Client Windows 12](#_Toc373067375)

[General organization 13](#_Toc373067376)

[View 13](#_Toc373067377)

[View Model 15](#_Toc373067378)

[Model 15](#_Toc373067379)

[Network Module 16](#_Toc373067380)

[Login Module 17](#_Toc373067381)

[Profile Module 18](#_Toc373067382)

[Friend Module 19](#_Toc373067383)

[Home Module 20](#_Toc373067384)

[CreateEvent Module 21](#_Toc373067385)

[ModifyEvent Module 21](#_Toc373067386)

[CreateRoom Module 22](#_Toc373067387)

[Rooms Module 22](#_Toc373067388)

[Room Module 23](#_Toc373067389)

[Create Deck module 23](#_Toc373067390)

[Game Module 23](#_Toc373067391)

[Client IOS 24](#_Toc373067392)

[VIEW: 25](#_Toc373067393)

[VIEW CONTROLLER: 26](#_Toc373067394)

[MODEL: 26](#_Toc373067395)

[Module Network: 27](#_Toc373067396)

[Module Login: 29](#_Toc373067397)

[Module Profile: 30](#_Toc373067398)

[Module CreateEvent: 30](#_Toc373067399)

[Android Client 31](#_Toc373067400)

[Network 32](#_Toc373067401)

[Module Login 32](#_Toc373067402)

[Module Event 33](#_Toc373067403)

[Module Profile 34](#_Toc373067404)

# Introduction

Magic Tactil is a Client / Server application, three clients are developed:

* Client Windows
* Client Android
* Client IOS

# The server

When the server starts, a thread will allocated to handle the main room. This room will contain every single user.

The aim of this thread is to accept the connection to the server. In this thread, for each client who is going to be connected a new thread will be allocated.

## The network module

This module permits to read Packet. When Packet is red, Packet is going to be treated and the right module will use the data he does have.

For each packet the server receives, information will be sent. This information will contain the "answer" of the user's request.

## The notification module

It permit to reorganize the data which enter and leave with a norm we had done (Packet structure)

If the data is leaving, the following modules will call it; otherwise, the network module will call it.

- Functions

Those modules are linked to the interpretation module and to the database. Each module will make a request to the database and will get his answer.

He permits the user to send message. Magic Tactil handles two types of sending:

- Private message: nickname of the person who ask, nickname of the person he wants and the message.

- Room message: nickname of the person who ask, the name of the room and the message. If the user is not connected or he is not in a room, an error will be returned.

Those two types of messages will be used to notify a user from an outside event which concern the user. This outside event can be a friend who has just been online or another player who has joined the room where the player is.

## The interpretation module

It permit to reorganize the data which enter and leave with a norm we had done (Packet structure)

If the data is leaving, the following modules will call it; otherwise, the network module will call it.

- Functions

Those modules are linked to the interpretation module and to the database. Each module will make a request to the database and will get his answer.

## Authentication module

Before everything else, the user has to create a Magic Tactil's account, for that, he needs to give the following information: name, surname, email and password. But he can also give more details like: age, gender, and location.

If the user wants to log in, the packet has to contain the following information: pseudonym/email and password.

If some information is wrong, an error will be returned.

## Profile module

This module permits to get the information about a user.

The needed data are: nickname of the person who ask, nickname of the person he wants

If they are different, the information will be sent depends on the wish of the user.

If they are equal, everything will be sent except the password.

If he wants to, the user can change his personal information.

In the case that the nickname is wrong, an error will be returned.

## Room module

As we said before, when an user is connecting, he is automatically pushed into the main room.

From this room, he can do:

- Create a room with: nickname and the name of the room

- Join a room with: nickname and the name of the room

From this new room he can do:

- Leave the room with: nickname and the name of the room

- Destroy the room with: nickname and the name of the room and has to be the owner.

If the new room is destroyed, every single user will pushed automatically in the main room.

## Chat module

He permits the user to send message. Magic Tactil handles two types of sending:

- Private message: nickname of the person who ask, nickname of the person he wants and the message.

- Room message: nickname of the person who ask, the name of the room and the message.

If the user is not connected or he is not in a room, an error will be returned.

## Cards module

This module permit to find cards, information the module needs are the characteristics of one card, the fields are: name, color, manacost, type, points (strength and defense), text and loyalty.

If the module does not find the card, an error will be returned.

## Deck module

This module permits to handle all decks. It's important to understand that when a player wins a card, it will be automatically pushed into the Main deck.

It's possible to create two types of deck, the first one is the "real deck" which contains only cards the player owns, and the other one is the "wish deck" which can contain any card even if he does not own it.

Each time a card will be pushed into a new deck it will be removed from the main deck.

If it's the opposite, it will be pushed into the Main deck.

The player can manipulate cards from the main deck like:

- Create a new deck: nickname, deck name, type of deck

- Add card to deck: nickname, deck name, ID card, number

- Remove a card to deck: nickname, deck name, ID card, number

## Shop module

This module permits to exchange cards or packet of cards between players and Magic Tactil.

We provide 3 types of selling:

- Sell one card: nickname of the buyer and the ID card

- Sell a packet of cards: nickname of the buyer, the name of the edition and the type ("packet")

For this kind of selling, the server will generate 15 cards randomly, but they have to respect one rule which is to have at least 1 rare card, 3 uncommon and 12 common. It is possible to get a mythic card instead of the rare.

- Sell a box of packet: nickname of the buyer, the name of the edition and the type ("box")

This follows the same way of doing but repeated 36 times. But the player has to get at least 4 mythic in his box.

In the case player prefers to exchange card between them, Magic Tactil provide a bid system:

- Classic bid ("EC")

- Buying directly ("AC")

The user can do:

- Put a bid on a card: nickname, name card, edition, time, price, type of bid

- Bid a card: nickname, id bid, price

- The Packet

The Packet structure his our way of communicating.

It contains:

- Source

- Destination

- Function code (4letters)

- Data

- Packet's data

The data from the Packet must be built in a special way which is:

- key\rvalue\nkey2\rvalue\n

In the case of an error, the data will be "KO"

## How does it work?

The following diagram shows how the server's behavior when he receives a request from a client.



When the server receives a connection request, the server behaves differently.

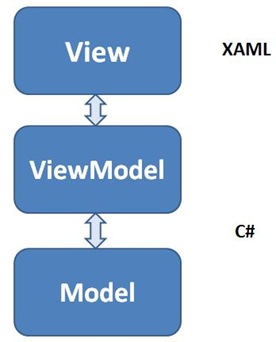


We have seen that the server creates a thread to the right client.

# Client Windows

The Windows client is compatible with Windows 7 and Windows 8. It uses the Surface 2 SDK.

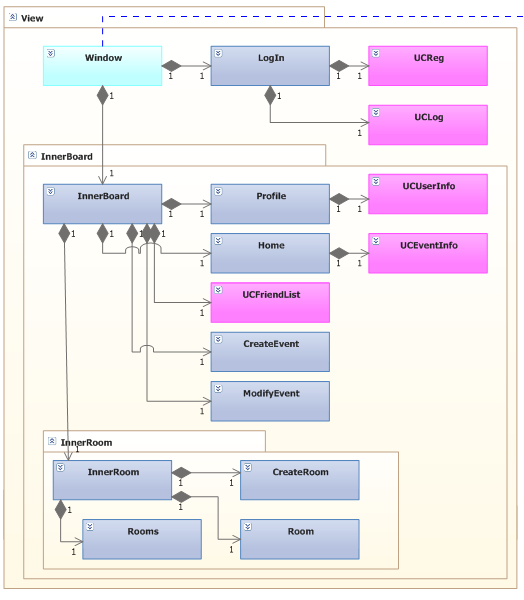
A MVVM pattern (~ MVC) was established. The View the part is in XAML, ViewModel and Model parts in C#, the model actually manages the communication with the server.



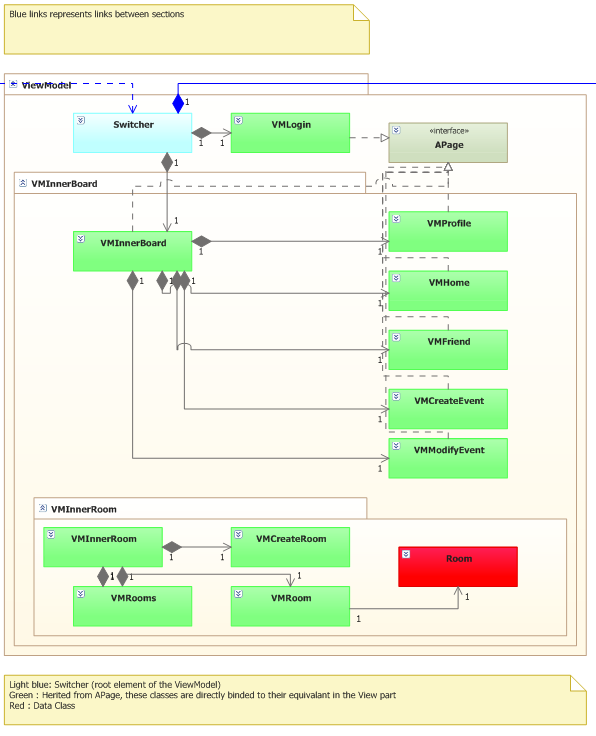
*Illustration: Pattern MVVM*

## General organization

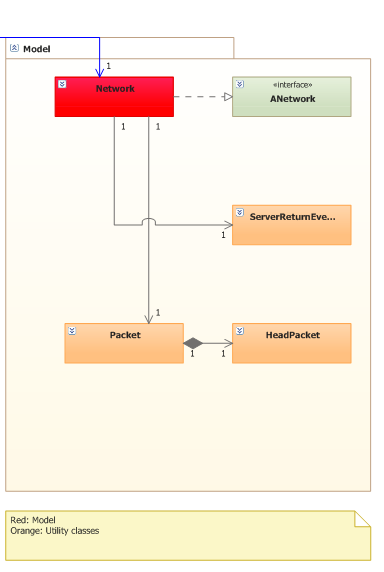
### View



### View Model



### Model

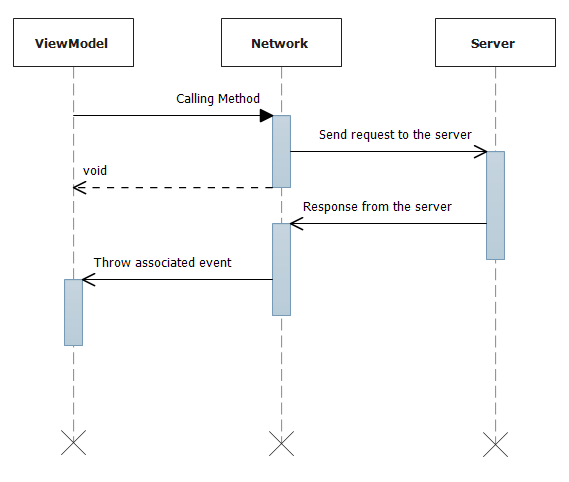


## Network Module

The Network module is the module that communicates with the server. This module and the View Model part form an Observer pattern. This allows asynchronous return requests to the server.

The Network module has a thread for sending messages to the server and a thread for receiving messages from the server.

For each possible request to the server, there is in the Network Module to a method and associated Event. When the View Model calls a Network module's method, the sending thread makes the request to the server. Then when the receiver thread gets the server's response, he throws the associated event.



*Illustration of the Observer pattern.*

View Model side, the module that uses the Network needs to subscribe to the associated event.

## Login Module

This module is used log in the server, log out or create an account. Login module is composed of VMLogin class (C #) in the Model View and User LogIn (xaml) interface in the View. The two entities are linked by bindings.

|  |  |  |  |
| --- | --- | --- | --- |
| VMlogin(C#) | LogIn(xaml) | View Model utility | View Utility |
| Boolean \_\_RBlogIn | RadioButton radioLogin | Call the login method or the method of creation of account based on the value. | Displays the login form or create an account (register) depending on the value. |
| String name | Textbox name | Used to communicate with the model | Allows the user to write his username. |
| String password | PasswordBox password | Used to communicate with the model | Allows the user to write his password. |
| String tmppassword | PasswordBox confirmPassword | Used to verify that there is no password error when creating account | Allows the user to enter his password a second time during account creation. |
| String email | TextBox email | Used to communicate with the model when creating account | Allows the user to write his email during account creation. |
| Action submit | Button Submit | Invokes the method of Model | Allows the user to confirm their information and connect or create an account |

## Profile Module

This module is used to edit personal information. The profile module is composed of VMProfile class (C #) in the Model View and Profile user interface (xaml) in the View. The two entities are linked by bindings.

Changing the values ​​of the interface causes the call server functions.

|  |  |
| --- | --- |
| VMProfile(C#) | Profile(xaml) |
| String username | Label/TextBox username |
| String mail | Label/TextBox mail |
| String firstName | Label/Textbox firstname |
| String surname | LabelTextBox 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 |

Before being transferred to the new model inputs are checked to meet the standards of the server.

## Friend Module

This module is used to manage a list of friends. The friend module consists of VMFriend class (C #) in the Model View and UCFriendList user control (xaml) in the View. The two entities are linked by bindings.

|  |  |  |
| --- | --- | --- |
| VMFriend(C#) | UCFriendList(xaml) | Utility |
| Boolean frORbl | RadioButton frORbl | Displays the list of friends or blacklist |
| ObservableCollection<String> frORblList | ListBox friendListBox | Contains black friends list or the list based on the value of frORbl |
| String friend | TextBox friend | The user can enter the name of a friend add |
| Action addFriend | Button addFriend | Adds the 'friend' in the friends list or the black list based on frORbl |

## Home Module

This module displays a list of events and information. The Home module consists of VMHome class (C #) in the Model View and user Home (xaml) interface in the View. The two entities are linked by bindings.

|  |  |  |
| --- | --- | --- |
| VMHome(C#) | Home(xaml) | Utility |
| ObservableList<String> events | ListBox events | The names of events on the server. |
| String eventName | TextBox eventName | The name of the selected event. |
| String creatorName | TextBox creatorName | The name of the creator of the event selected. |
| String date | TextBox date | The date of the event selected |
| String location | TextBox location | The location of the selected event |
| String content | TextBox content | The description of the selected event |

## CreateEvent Module

This module allows you to create an event. The CreateEvent module consists of VMCreateEvent class (C #) in the Model View and user CreateEvent (xaml) interface in the View. The two entities are linked by 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 |

## ModifyEvent Module

This module allows you to change an event which one is the creator. The module consists of VMModifyEvent class (C #) in the Model View and user ModifyEvent (xaml) interface in the View. The two entities are linked by 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 |

## CreateRoom Module

This module allows to create a room. The module consists of VMCreateRoom class (C #) in the Model View and user CreateRoom (xaml) interface in the View. The two entities are linked by bindings.

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

## Rooms Module

This module allows to consult the existings rooms and join one. The module consists of VMRooms class (C #) in the Model View and user Rooms (xaml) interface in the View. The two entities are linked by bindings..

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

## Room Module

This module allows to leave a room. The module consists of VMRoom class (C #) in the Model View and user Room (xaml) interface in the View. The two entities are linked by bindings.

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

## Create Deck module

This module allows to create decks. The module consists of VMInnerDecks class (C #) in the Model View and InnerDeck (xaml) interface in the View. The two entities are linked by bindings.

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

## Game Module

This module allows to plays. The module consists of VMGame class (C #) in the Model View and Game (xaml) interface in the View. The two entities are linked by bindings.

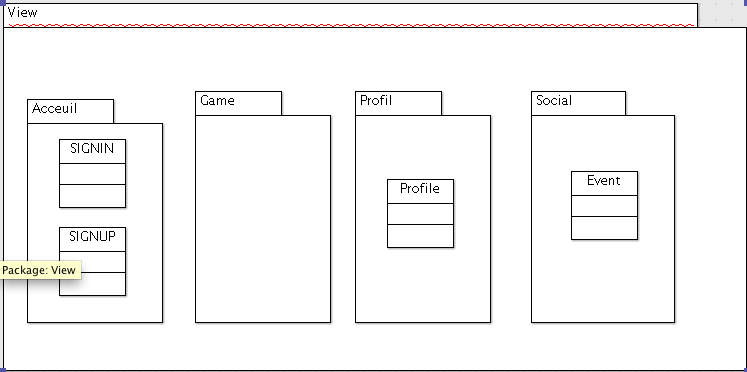
|  |  |
| --- | --- |
| VMGame(C#) | Game (xaml) |
| List<card> hand | Surfacelistbox hand |
| List<Card> deck | SurfaceListBox deck |
| List<Card> field | SurfaceListBox field |
| List<card> graveyard | Surfacelistbox graveyard |
| List<card> exile | Surfacelistbox exile |

# Client IOS

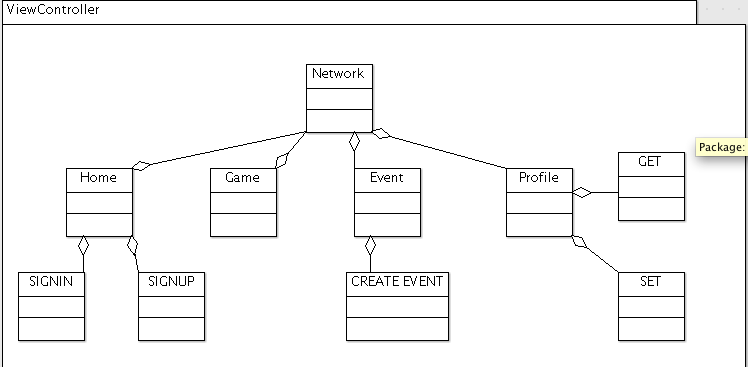
IOS client is compatible with iPhone (3G, 3GS, 4, 4S, 5) iPad (1, 2, Retina, Mini) and iPod Touch. It is developed with the official Apple technologies in Objective-C with Xcode 4.6.2. It will at least have IOS 6.0 on his iDevice to run the application.

A design pattern MVC (Model View Controller) has been implemented. The view is generated in XML, the viewController and the model is in Objective-C. The viewController manages the Events view (touch, multi-touch, scroll, etc ...) and the model is the bridge between the application server.

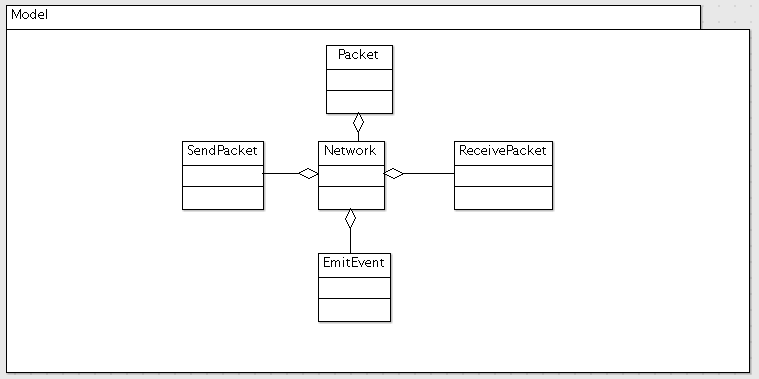
## **VIEW:**



## VIEW CONTROLLER:



## MODEL:



## Module Network:

The Network module allows asynchronous way to send and receive packets to the server. It uses a standard TCP connection. The Network module receives the packets in real time and sends the appropriate signals to wake up the corresponding modules. It uses a producer-consumer system with a tail that is supplied by the server and used by the corresponding modules.

## Module Login:

|  |  |  |  |
| --- | --- | --- | --- |
| VClogin(Objective C) | SIGNIN(xml) | ViewController Utility | View Utility |
| UITabBard | UIButton | Call the login method or the method of creation of account based on selected button | Displays the login form or account creation. |
| NSString name | UITextField name | Used to communicate with the model | Allows the user to write his username. |
| NSString password | UITextField password | Used to communicate with the model | Allows the user to write his password. |
| NSString Confirmpassword | UITextField confirmPassword | Used to verify that there is no password error when creating account | Allows the user to enter his password a second time during account creation. |
| NSString email | UITextField email | Used to communicate with the model when creating account | Allows the user to write his email during account creation. |
| NSEvent submit | UIButton Submit | Call method of Model | Allows the user to confirm their information and connect or create an account |

Login module can register, connect or disconnect from the server. It consists of classes and signin SIGNUP which are each associated with a view.

## Module Profile:

The profile module allows seeing all the account information; it also allows you to edit its information. It is composed of the Profile class that is associated with a view.

|  |  |
| --- | --- |
| VCProfile(Objective C) | Profile(XML) |
| NSString username | UILabel/UITextField username |
| NSString mail | UILabel/UITextField mail |
| NSString firstName | UILabel/UITextfield firstname |
| NSString surname | UILabelUITextField 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:

This module allows you to create an event. This module is composed of Classes Events, EventObject, AddEvent which are each associated with a view.

|  |  |
| --- | --- |
| 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 |

# C:\Users\Benjamin\Pictures\bugdroid.pngAndroid Client

To run the Android application MagicTactil we need at least the API version 8.

All phones running Android version 2.2.x (Froyo) and more can use MagicTactil.

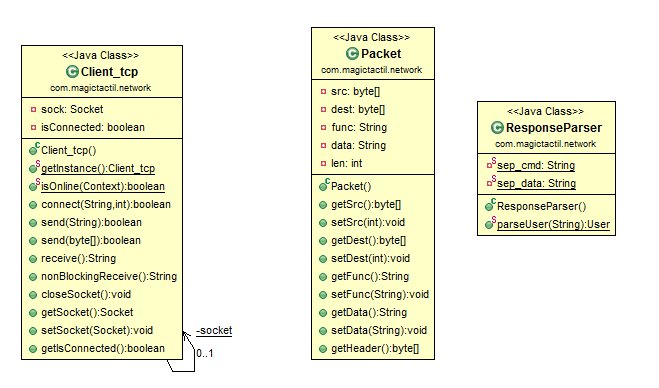
A tablet version of the application is available.

The design pattern MVC (Model View Controller) is used. The views (layouts) are in XML. The models and the controllers are in JAVA.

The library used for this project are :

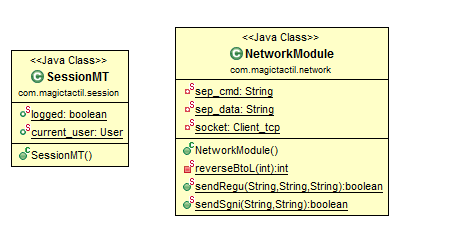
* ActionBarSherlock : allow the use of the action bar in all android versions
* SDK Facebook : Allow using Facebook features.

## Network



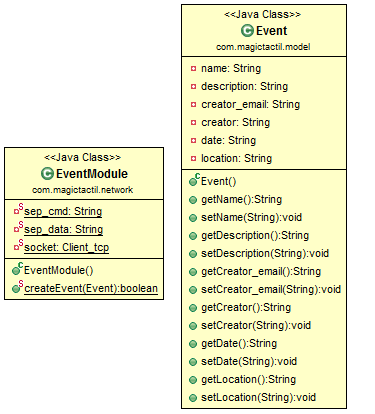
## Module Login

The login part allow the sign in to an existing account and a sign up to the server.



## Module Event

The event part permit to create new events on the server.



## Module Profile

The profil part permit to see the profil informations and to edit it.

