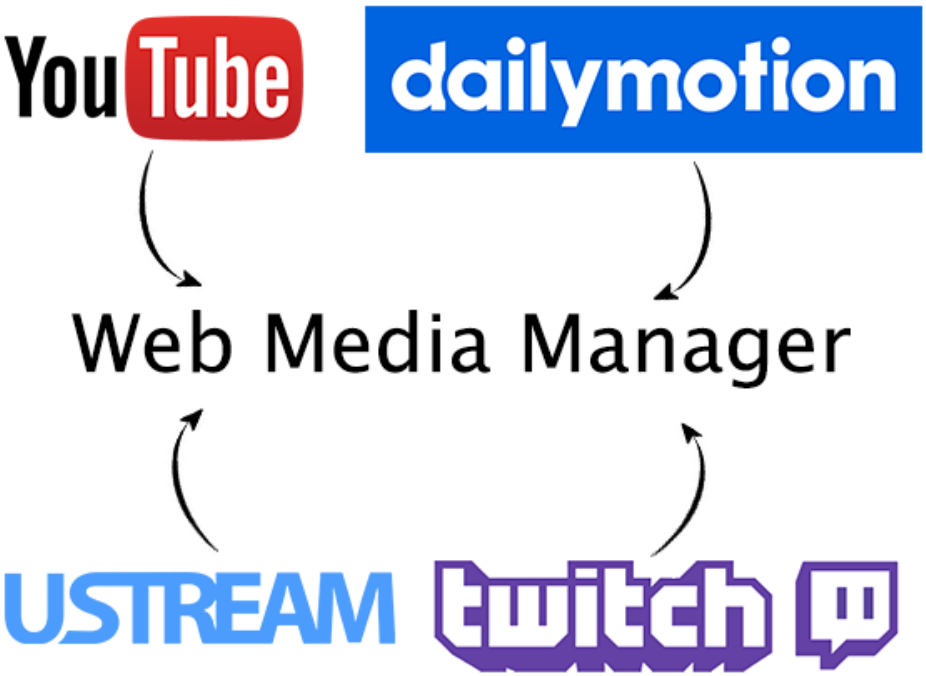


Web Media Manager

Mémoire

CFPT Informatique
Jean-Philippe Froelicher
T.IS-E2A

Superviseur : Christophe Maréchal

2014 - 2015

Table des matières

Chapitre 1

Résumés

1.1 Français

Web Media Manager est une application qui centralise les services des sites de vidéos tels que : "Twitch" ; "Youtube" ; "Dailymotion" ; "Vimeo" etc. L'application doit en grande partie pouvoir remplacer les différents sites. L'application doit également pouvoir être évolutive et donc pouvoir centraliser n'importe quel site vidéo qui propose une API. L'application est codée en C#, il y a une interface principale qui regroupe toutes les diffusions de flux vidéo en direct et toutes les dernières vidéos sorties. L'application offre également la possibilité de pouvoir naviguer dans le contenu de chaque site séparément. Toutes les fonctionnalités principales qu'offrent les sites sont présentes dans l'application. Afin d'obtenir les informations des différents sites, j'utilise les API qu'ils proposent. En effet, tous ces sites proposent une API REST qui me permet donc de récupérer les données de la même manière pour tous les sites. Afin d'avoir un format générique, j'ai créé une structure qui regroupe toutes les informations communes de tous les sites.

1.2 English

Web Media Manager is an application that centralizes services of video sites such as : "Twitch" ; "Youtube" ; "Dailymotion" ; "Vimeo" etc. She must largely be able to replace the different sites. The application must be scalable and therefore be able to centralize any video site that offers an API. It is coded in C #, there is a main interface that brings together all the live streams videos and all the latest video outputs. The application give the possibility to navigate through the content of each site separately. All the main features offered by the website are present in the application. In order to obtain the information from the different sites, I use the API they offer. Indeed, all websites offer a REST API that allows me to retrieve data in the same way for all sites. In order to have a generic format, I created a structure which includes all common information from all the sites.

Chapitre 2

Cahier des charges

2.1 Titre du projet

WebMedia Manager

2.2 Objectifs du projet

Création d'une application permettant l'utilisation des différents services proposés par les sites de vidéos et de diffusion de flux vidéo en direct. Les fonctions de bases liées aux services en question sont intégrées génériquement dans l'application. Des fonctions ajoutées par moi-même y sont également ajoutées. L'application a pour but de faciliter l'utilisation de ses services pour les utilisateurs ayant une fréquentation régulière de ceux-ci.

2.3 Description détaillée

L'application propose un certains nombre de service lié à la plateforme d'hébergement et de diffusion des vidéos. En effet, plusieurs sites vidéo comme Youtube, Dailymotion, Twitch etc. proposent des services de bases pour leurs utilisateurs. Elle reprend si possible dynamiquement les différentes fonctions de chaque site et les proposes dans une interface créée à cet effet.

Les fonctions de bases que l'application propose pour chaque site :

- Connexion avec un compte lié au site, donc créer auparavant ;
- Modification des différents paramètres de comptes ;
- Recherches de flux vidéos suivant différents critères : nom d'un flux vidéo, nom du jeux, nom d'une chaîne, nom d'un utilisateur, etc ;
- Affichage d'une vidéo ou d'une diffusion en direct ;
- La fonction "Suivre" ou "S'abonner" qui consiste à être mis au courant des nouvelles vidéos / diffusion ;

- Affichage détaillé d'un utilisateur : accès à ses informations publique, ses vidéos etc ;
- Affichage des vidéos / diffusions en direct les plus populaires ;
- Affichage de l'espace communautaire : commentaire vidéos, Chat sur une diffusion en direct

Les fonctions ci-dessus sont celle de base pour chaque site à quelques exceptions près.
Ci-dessous, les fonctions ajoutées par moi-même dans l'application :

- Système de notification lorsqu'une vidéo sort ou qu'une diffusion en direct commence ;
- Création de catégorie pour organiser les différents flux vidéos suivis, les catégories sont inter-services, c'est-à-dire que l'on peut mélanger les différents contenu des sites.
- Création de playlist pour lire plusieurs vidéos à la suite, également inter-services. Par contre, cela ne s'applique uniquement sur les vidéos, on ne peut pas créer de playlist de diffusion en direct.

Pour la réalisation de cette application, j'utilise le langage C#.

2.4 Inventaire des étapes

Début : Lundi 13/04/2015
Reddition intermédiaire (doc + poster) : Vendredi 30/04/2015
Reddition finale : Lundi 01/06/2015

2.5 Inventaire du matériel

PC + 2 écrans

2.6 Inventaire des logiciels

Visual Studio 2013 Professionnel

2.7 Délivrables (documents à restituer)

- 1 journal de bord (format A5)
- 1 poster A2
- 2 exemplaires papier de la documentation technique
- 2 exemplaires papier du mode d'emploi (si besoin)
- 1 CD/DVD ROM contenant tous les fichiers (sources + documentation + poster)
- Une démonstration fonctionnelle du projet, une solution parmi :
  - live CD, live USB
  - machine virtuelle (VirtualBox) pré-configurée

2.8 Éléments mesurables (servant à l'évaluation)

Réalisation des objectifs, mesurées selon la grille d'évaluation.

Chapitre 3

Étude d'opportunité

3.1 Introduction au projet

Le but de ce projet est de réaliser une application permettant l'utilisation des différents services proposés par les sites de vidéos et de diffusion de flux vidéo en direct.

3.1.1 Médias vidéo web

Aujourd'hui, les sites comme Youtube, Dailymotion, Twitch qui propose du contenu vidéos sont de plus en plus visités. Ses sites, ont tous un point commun, d'autres personnes mettent des vidéos en ligne pour divertir les spectateurs.
Le phénomène a pris une telle ampleur que certain "créateur de contenu audiovisuel" sont même payé par ses sites par rapport à leur popularité. De plus en plus de personnes, surtout les jeunes, passent leurs temps devant des vidéos et délaissent de plus en plus la télévision.

Direct

Les vidéos en direct sont de plus en plus présentes sur le web. En effet, grâce en grande partie aux jeux vidéos, le phénomène des personnes créant du contenu audiovisuel s'est également répandu sur du contenu en direct. Des sites comme Twitch ou Dailymotion proposent à ses utilisateurs de diffuser du flux vidéo en direct.
La plupart du temps, c'est pour les jeux vidéos, les diffuseurs jouent sur leurs ordinateurs / consoles puis, retransmet l'image sur le site. Ainsi, des personnes du monde entier ont la possibilité de regarder la partie de jeu d'une personne. Les spectateurs ont même la possibilité de discuter avec les autres spectateurs, et même des fois avec le diffuseur de contenu.
Les diffuseurs sont payés grâce aux dons de leurs communautés, certains font ça à plein temps et donc sont contraint à entretenir une grande communauté.

Différé

Les vidéos "différées" sont depuis un petit moment déjà présentent sur le web. En effet, le phénomène des vidéos sur le web n'est pas tout nouveau, mais, ce n'est que depuis peu que le phénomène à prit une grande ampleur.
Si un créateur de contenu a atteint une certaine popularité, il peut enfin prétendre à faire de l'argent avec ses vidéos. En effet, le premier site de vidéos du monde Youtube rémunère les créateurs de contenu.

3.1.2 Pourquoi avoir choisi ce sujet ?

J'ai choisi ce sujet, car je porte un réel intérêt au monde audiovisuel sur le web. En effet, je fais partie des jeunes qui a délaissé la télévision pour les vidéos sur internet. Il y a plusieurs sites de vidéos et souvent, il est difficile de s'y retrouver, l'application que j'ai pensée a pour but de faciliter la vie des personnes comme moi qui utilise ses sites régulièrement. Le fait que le monde de l'audiovisuel est en plein boum sur le web me stimule encore plus à l'idée de créer une application dans ce domaine.
On retrouve également de plus en plus de sites proposant à ses utilisateurs de pouvoir diffuser du flux vidéo en direct, ce qui consolide l'intérêt de mon application.

3.2 Analyse de l'existant

Il n'y a pas d'application similaire à celle que je vais réaliser. Tout de même, les différentes fonctions que je veux y intégrer sont déjà présentes sur les différents sites.

3.2.1 Existant

- Twitch now : Application Google Chrome
- Dailymotion Games : Application Android

3.2.2 Critique de l'existant

Twitch now : Cette application permet à l'utilisateur de naviguer dans les différentes diffusions de flux vidéo en direct de Twitch. C'est une application Google Chrome, elle permet de naviguer sur Twitch sans aller sur le site. Le système de "suivis" est celui de Twitch, c'est-à-dire qu'il faut se connecter avec son propre compte Twitch.

Dailymotion Games : Cette application permet aux utilisateurs de Dailymotion de naviguer entre les différentes diffusions de flux vidéo en direct. Par contre, on ne peut pas se connecter à son compte personnel pour avoir uniquement que les diffusions que l'on est abonné. Les différentes diffusions sont classées soit par vues, soit par jeux. Évidemment, les diffusions de flux vidéo peuvent être visionnée depuis l'application. Dailymotion Games est une application officielle de Dailymotion.

Chapitre 4

Analyse fonctionnelle

4.1 Description du fonctionnement

4.1.1 Fonctions des sites

Générique

Tous les sites proposent des fonctions "communes" de base qui sont intégrées à WebMedia Manager :

- Affichage des dernières vidéos / diffusion en direct
- Recherche une vidéo / diffusion en direct
- Affichage d'une vidéo / diffusion en direct
- Connexion au service en question
- Modification des paramètres du compte connecté
- Afficher les détails d'un utilisateur
- Afficher les vidéos / diffusion en direct populaire
- Gestion des notifications
- Suivre une chaîne / un utilisateur

Diffusion en direct

Les fonctions spécifiques aux sites qui proposent des diffusions en direct :

- *Chat* : Discuter avec les autres spectateurs
- Abonnement payant [1]

Les utilisateurs ont le la possibilité de faire des donations, mais pas par le biais des sites de diffusions.

Vidéo en différé

Les fonctions spécifique aux sites qui proposes des vidéos en différé :

- Commentaires : Donner un avis / discuter avec les autres spectateurs

1. Offre des avantages à l'abonné

4.1.2 Gestion des catégories

Les catégories servent à rassembler plusieurs vidéos de différents sites. En effet, grâce aux catégories, les vidéos peuvent être organisées, c'est-à-dire qu'on peut les retrouver facilement au même endroit, même si elles ne viennent pas du même site.

Action possible sur une catégorie :
- Ajouter
- Modifier le nom
- Supprimer

La seule action possible sur une vidéo est de la supprimer.

4.1.3 Gestion des listes de lecture

Les listes de lecture servent à lire un certain nombre de vidéos à la suite. Les listes de lectures sont également inter-site.

Action possible sur une liste de lecture :
- Ajouter
- Modifier le nom
- Supprimer

Actions possibles sur une vidéo :
- Supprimer
- Monter (dans la liste)
- Descendre (dans la liste)
- Déplacer à X position (dans la liste)

4.1.4 Notifications

Les notifications servent à être tenues au courant lors de nouvelle diffusion en direct ou lorsqu'une nouvelle vidéo sort. Lorsque un utilisateur suit un auteur de vidéo, il sera de toute façon notifié. L'application donne la possibilité à l'utilisateur d'activer ou de désactiver les notifications.

4.2 API (Application Programming Interface)

Les API des différents sites permettent de faire des interrogations entre les bases de données et l'application.

4.2.1 API REST

Les API REST [2] servent à accéder à une ressource par son URI grâce au protocole HTML pour procéder à diverses opérations : GET ; POST ; PUT ; DELETE.
Le format de représentation des données est libre, dans le cas des API que l'on utilise, le format est en JSON [3]. (cf. figure 4.1)

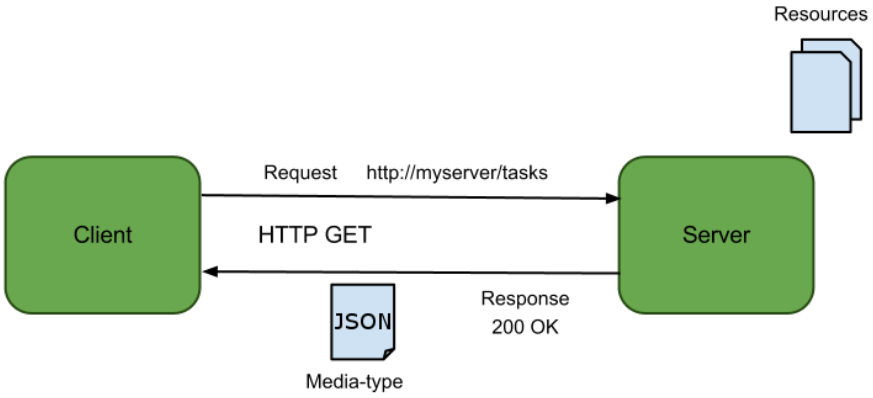


FIGURE 4.1: REST

Les ressources sont accessibles via les liens que l'API met à disposition :
- `https://api.dailymotion.com` : pour le site Dailymotion
- `https://api.twitch.tv` : pour le site Twitch
- `https://api.vimeo.com` : pour le site Vimeo

Toutes ses API mettent à disposition des URL qui servent à récupérer des ressources précises.

4.3 Lecteur vidéo

Afin d'afficher le flux vidéo des différentes vidéos ou des différentes diffusions en direct, les sites ont leur propre lecteur vidéo. Ils se présentent généralement sous un format *Flash player* ou HTML5.

2. Representational State Transfer
3. JavaScript Object Notation

4.3.1 Flash player

Le plus part des sites utilisent le lecteur flash pour leurs lecteurs vidéo. Le lecteur flash utilise le protocole RTMP [4], c'est un protocole réseau propriétaire développé par Adobe Systems. Il sert à la diffusion de flux de données en streaming(audio, vidéo ...) entre un serveur et un client.

4.3.2 HTML5

Un nouveau format de lecteur vidéo est apparu sur le web depuis peu, il s'agit du lecteur HTML5. Ce lecteur utilise un autre codec (VP9), le codec libre de Google qui compresse la vidéo. Ce codec a été mis en place surtout pour accélérer l'accession au contenu HD 4k à 60 fps [5]. Le fait d'utiliser le lecteur HTML5 est un plus dans le multi-plateforme.

4.4 Outils communautaires

Les différents sites de vidéos mettent à disposition des utilisateurs des outils afin de pouvoir communiquer avec d'autre membre du site ou directement s'adresser à l'auteur de la vidéo.

4.4.1 Chat IRC

Le *chat* est utilisé le plus souvent lors de diffusion de flux vidéo en direct. Afin d'avoir un contact direct avec le diffuseur, il faut avoir un support sur lequel le diffuseur peut lire rapidement.

La plupart des *chat* utilisé pour les diffusions de flux vidéo en direct sont des *chats* IRC [6].

IRC est un protocole de communication textuelle, il sert à la communication instantanée sous la forme de discussions de groupe par l'intermédiaire de canaux de discussion. Il peut également être utilisé pour communiquer d'un à un. Le principe est que chaque utilisateur est connecté à un serveur IRC qui eux-mêmes sont relié avec d'autres serveurs IRC. Ainsi, toutes les personnes peuvent discuter sur des forums publics ou privés.

Les différents sites de diffusions en direct utilise le client IRC "Kiwiirc" conçue spécialement pour le web, cela ajoute au chat IRC plein de caractéristiques qui améliore l'utilisation du chat.

4.4.2 Commentaires

Les commentaires servent à discuter avec d'autre personne sur la vidéo en question. Ils servent également à communiquer avec l'auteur de la vidéo. Il n'y a pas de commentaire sur les diffusions de flux vidéo en direct.

4. Real Time Messaging Protocol
5. Frame Per Second
6. Internet Relay Chat

4.5 Connexion

Afin de se connecter aux différents services avec un compte personnel, les API utilisent le protocole OAuth2.

4.5.1 OAuth2

La plus part des grands sites utilisent le protocole OAuth2 pour l'authentification au compte personnel, en effet, ce protocole permet d'obtenir un accès limité à un service via HTTP par le biais d'une autorisation. La demande d'accès est demandée par le client, en l'occurrence WebMedia Manager. (cf. figure 4.2)
OAuth2 définit 4 rôles :

- Détenteur des données (l'utilisateur)
- Serveur de ressources (Twitch, Youtube, Dailymotion ...)
- Client (WebMedia Manager)
- Serveur d'autorisation (Twitch, Youtube, Dailymotion ...)

Token (jeton)

Lorsque le client fait une demande d'authentification, le serveur d'autorisation délivre un *token*. Un *token* permet au serveur de ressources d'autoriser la mise à disposition des données d'un utilisateur. Il a une durée de vie limité qui est définie par le serveur qui délivre les *tokens*. Il doit rester le plus confidentiel possible, même l'utilisateur ne voit pas son *token* attribué.

Scope (portée)

Le *scope* est un paramètre qui sert à définir les droits sur un *token*. En effet, le serveur d'autorisation propose une liste de *scope* et lors de l'authentification, attribut ses droits sur le token.

Type d'autorisation

Il existe deux types d'autorisation : Flux de code d'autorisation(*Autorization Code Flow*) et l'autorisation implicite(*Implicit Grant Flow*). L'autorisation implicite s'utilise quand l'application se trouve côté client. La demande d'authentification se fait de la sorte :

1. L'application souhaite accéder aux données
2. Requête d'autorisation au serveur
3. Si l'accès est autorisé, le serveur d'autorisation délivre le token
4. Utilisation du token pour certaine requête

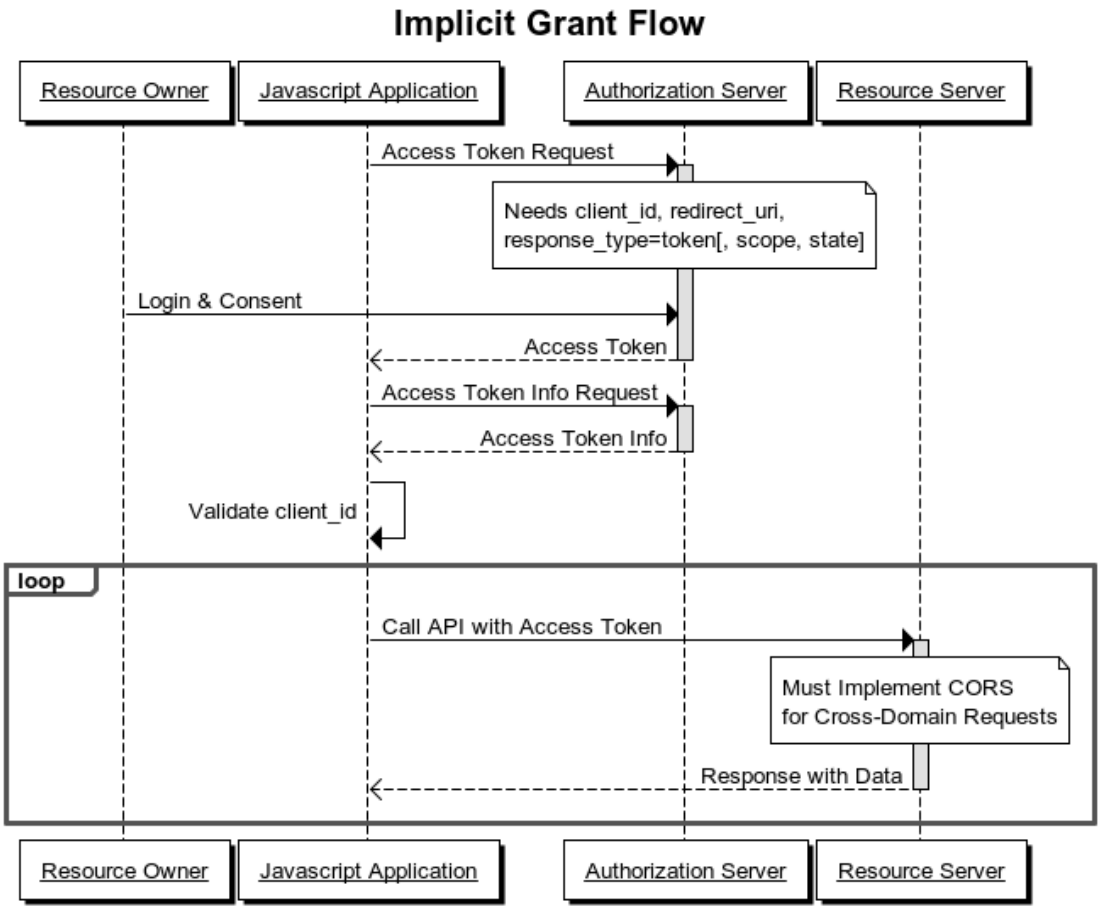


FIGURE 4.2: Schéma OAuth2 : Droit implicite

4.6 Interface homme-machine

4.6.1 Miniature

Les miniatures des vidéos sont affichées sous cette forme. (cf. figure 4.3)

FIGURE 4.3: Miniature

4.6.2 Notification

Les notifications sont affichées sous cette forme. (cf. figure 4.4)

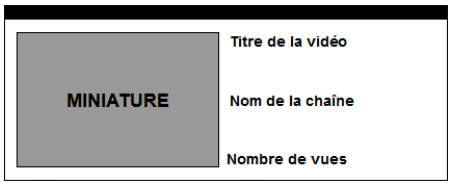


FIGURE 4.4: Interface notification

4.6.3 Navigation

Les couleurs représentent les différentes fenêtres de l'application. Les flèches rouges marquent le changement du menu de gauche dans l'interface graphique. (cf. figure 4.5)

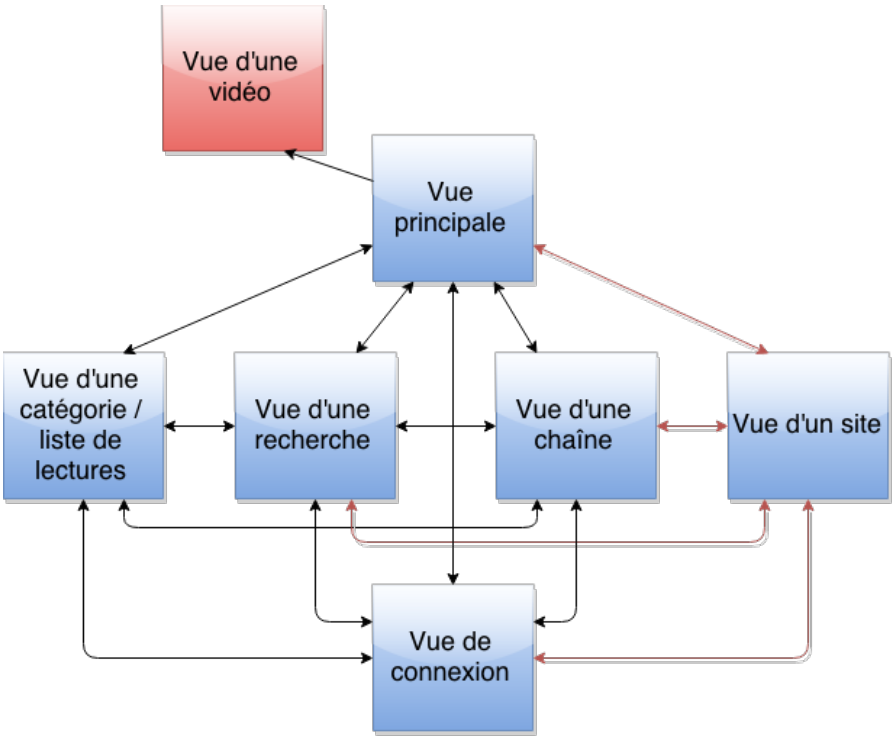


FIGURE 4.5: Navigation de l'application

4.6.4 Interface personnelle

Si l'utilisateur de l'application n'est connecté à aucun compte, le contenu de cette interface se désactive sauf la partie 1 qui permet à l'utilisateur de se connecter avec ses différents comptes. (cf. figure 4.6)

1. Boutons permettant d'accéder à l'interface concernant le site en question
2. Liste des catégories et des listes de lecture et affiche le contenu dans l'interface
3. Recherche de flux vidéo et affiche le résultat dans l'interface
4. Liste des diffusions de vidéo en direct et ouvre une nouvelle interface pour voir le flux vidéo
5. Liste des vidéos dernièrement sorties et ouvre une nouvelle interface pour voir le flux vidéo

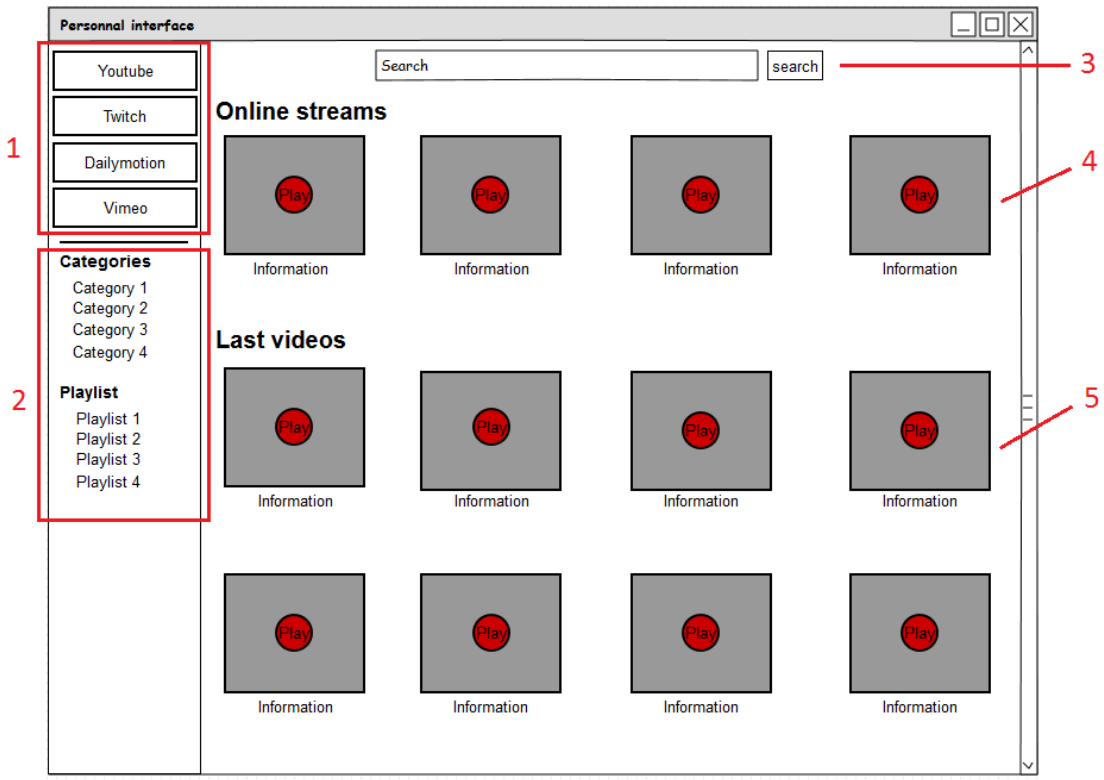


FIGURE 4.6: Interface personnel

4.6.5 Interface des sites

L'interface des sites est commune pour tous les sites. Chaque site à une interface générique. (cf. figure 4.7)

1. Bouton permettant d'accéder à l'interface personnel
2. Les options possibles concernant le compte de l'utilisateur
3. La liste des abonnements / suivis
4. Recherche de flux vidéo et affiche le résultat dans l'interface
5. Liste des flux vidéos

FIGURE 4.7: Interface des sites

4.6.6 Interface vidéo

L'interface vidéo est générique, pour chaque vidéo de chaque site, c'est la même interface. (cf. figure 4.8)

1. Flux vidéo
2. Nom de la vidéo
3. Ajouter le flux vidéo à une catégorie ou liste de lecture
4. S'abonner à l'auteur de la vidéo
5. Informations concernant la vidéo
6. *Chat* cf. 4.4.1 ou espace commentaires
7. Nombre de vues de la vidéo

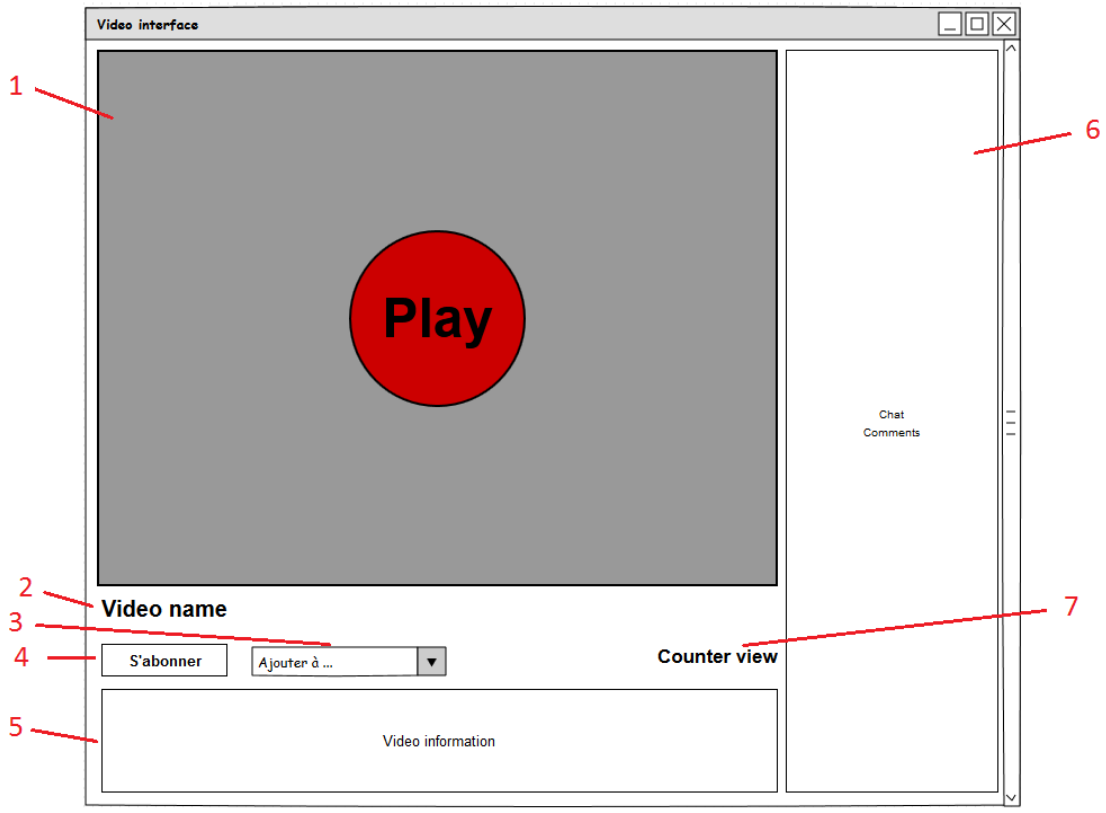


FIGURE 4.8: Interface d'une vidéo

4.6.7 Interface d'une chaîne

(cf. figure 4.9)

1. Logo et bannière de la chaîne en question
2. Bouton pour afficher les suivis et suiveurs de la chaîne
3. Filtre sur les vidéos à afficher : Populaire, nouvelles etc ...
4. Liste des flux vidéos
5. Diverses informations concernant la chaîne

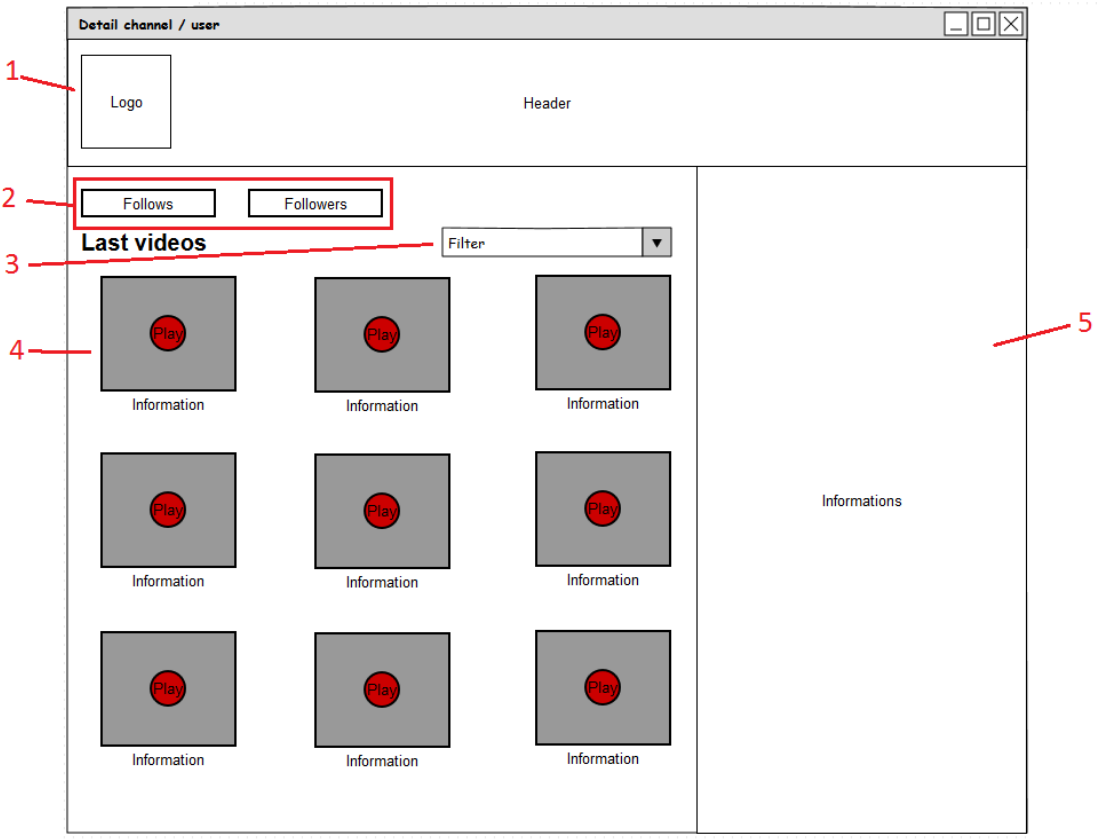


FIGURE 4.9: Interface d'une chaîne / utilisateur

4.6.8 Interface de connexion

(cf. figure 4.10)

La page de connexion est affichée lorsque l'utilisateur clique sur le bouton d'un site sur lequel il n'est pas encore connecté. La page web se charge suivant sur quel site l'utilisateur se connecte.

FIGURE 4.10: Interface de connexion

Chapitre 5

Analyse organique

5.1 Généralité

5.1.1 Langage en environnement de développement

L'application est développée en C# sur Microsoft Visual Studio 2013.

5.1.2 Format des fichiers et enregistrements

L'application stocke sa configuration sous forme de fichier. En effet, l'application a besoin d'enregistrer les catégories, listes de lecture ainsi que tout leurs contenus. Elle a également besoin d'enregistrer quelques configurations. C'est pour cela que j'utilise le format de fichier INI [1] car il est très facile à manipuler. (cf. figure 5.1)

```
[categorie1]
link=lien1;lien2;lien3;lien4;lien5;lien6;lien7
[categorie2]
link=lien1;lien2;lien3;lien4;lien5;lien6;lien7,lien8
[categorie3]
link=lien1;lien2;lien3
```

FIGURE 5.1: Exemple de fichier INI

Pour plus de faciliter dans l'enregistrement des données, je crée deux fichiers INI : Category.ini et Playlist.ini

1. Fichier de configuration

5.1.3 SVideo & SChannel

Afin d'avoir un type générique pour traiter les vidéos de tous les sites de la même manière, j'ai créé deux structures. Elles contiennent toutes les informations commune de touts sites.

SVideo : structures pour les vidéos et vidéos en direct. (cf. figure 5.2)

```
public struct SVideo
{
    public string id;
    public string videoName;
    public string channelName;
    public int nbViews;
    public string description;
    public string preview;
    public string playerLink;
    public DateTime createdAt;
    public string link;
    public bool live;
    public string siteName;
    public string url_irc;
    public bool channelIsFollowed;
}
```

FIGURE 5.2: Structure SVideo

SChannel : structure pour les chaînes. (cf. figure 5.3)

```
public struct SChannel
{
    public string id;
    public string channelName;
    public string description;
    public string logoLink;
    public string headerLink;
    public int nbTotalViews;
    public int nbFollowers;
    public DateTime createdAt;
    public string siteName;
}
```

FIGURE 5.3: Structure SChannel

5.2 Conception

5.2.1 Modèle

StreamingSite

Pour pouvoir traiter les différents sites de la même manière, j'ai créer une classe de base qui sert de "modèle" aux classes des sites. Les classes de sites doivent contenir les mêmes méthodes que la classe de base. (cf. figure 5.4)

FIGURE 5.4: StreamingSite table UML

Pour ajouter d'autres sites, il faut juste créé une classe qui respecte la classe de base. Les méthodes communes à touts les sites sont virtuelles, en effet, étant donné que pour chaque site les traitements ne sont pas les mêmes, il faut réécrire la méthode pour chaque site dans leurs classes respective. (cf. figure 5.5)

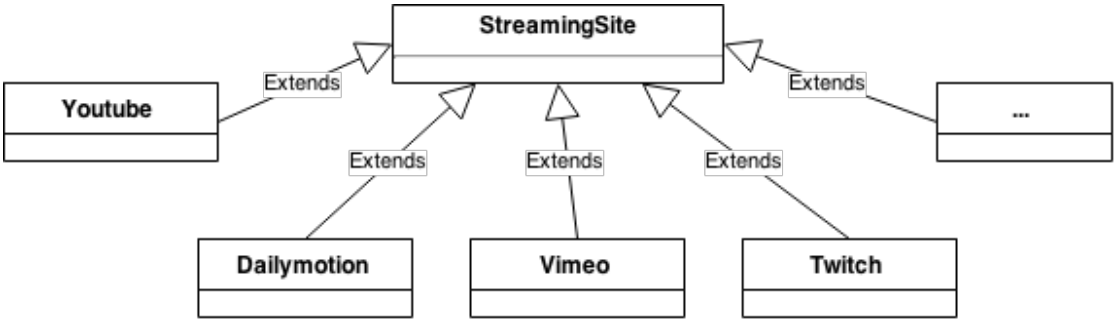


FIGURE 5.5: Diagramme StreamingSite

5.2.2 Catégorie et liste de lectures

Pour différencier les catégories et les listes de lectures, j'ai créé une classe de base *Container* qui contient toutes les méthodes concernant les deux types de container. Étant donné que les listes de lectures sont des catégories avec une gestion de lecture en plus, j'ai créé une classe *Playlist* qui contient uniquement les méthodes concernant les listes de lectures. (cf. figure 5.6)

Les vidéos sont stockées sous forme de liens, cela me permet de reconnaître de quel site est la vidéo (cf. figure 5.1). Ainsi, je peux récupérer les informations d'une vidéo grâce à l'id de la vidéo qui se trouve dans son lien.

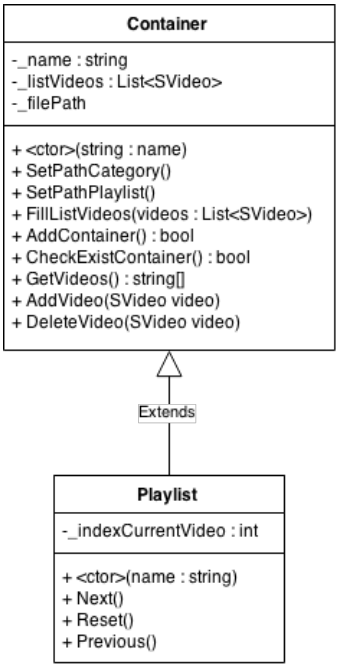


FIGURE 5.6: Diagramme Container

5.2.3 Authentification

L'authentification se fait au niveau du site, elle est différente pour chaque site. (cf. figure 5.7) Cette classe contient toutes les informations indispensables pour la connexion OAuth2. (cf. 4.5.1)

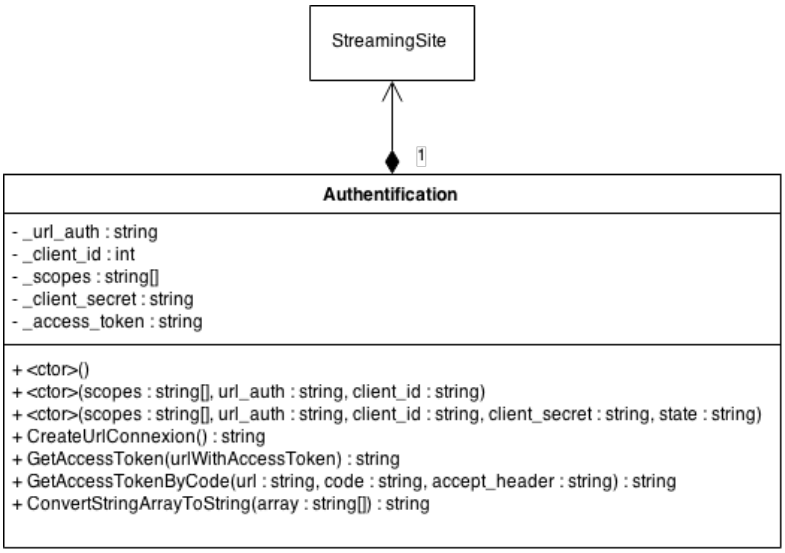


FIGURE 5.7: Authentification table uml

5.2.4 Global

Cette application est MVC. (cf. figure 5.8)

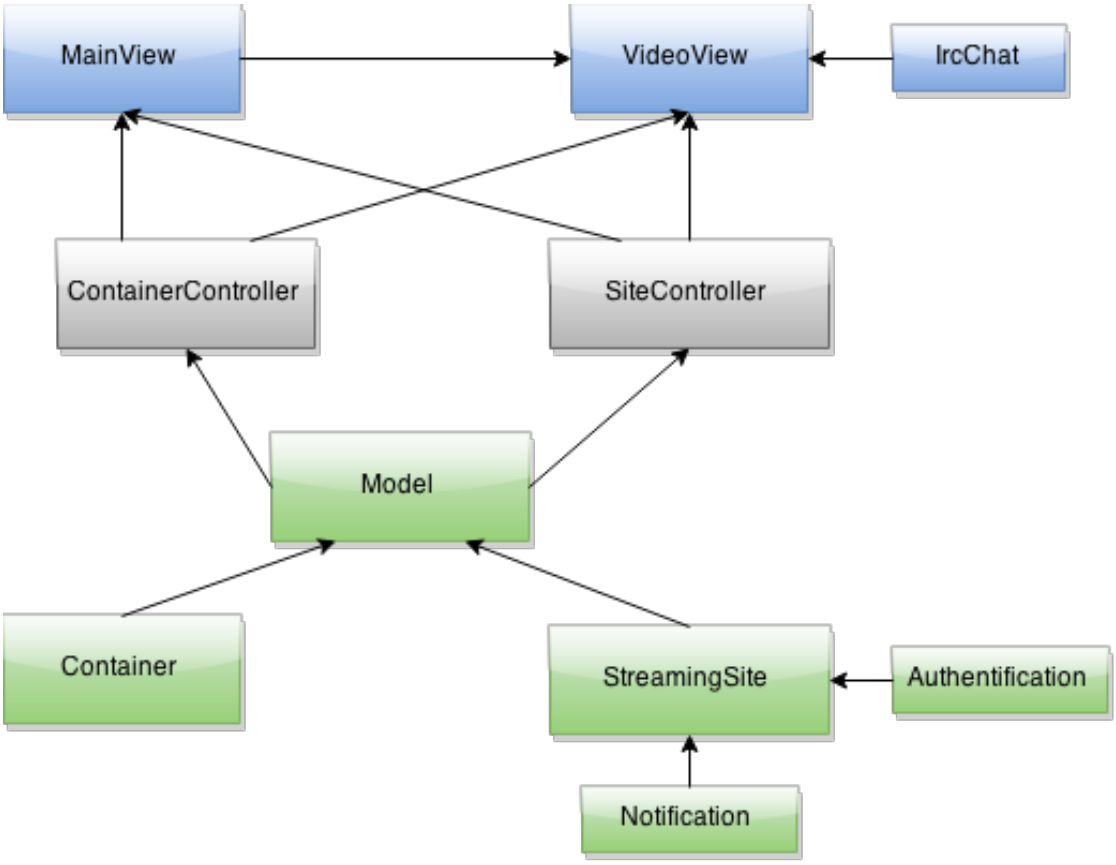


FIGURE 5.8: Diagramme global

5.3 Outil communautaire

5.3.1 IRC

Le chat IRC (cf. 4.4.1) a été réalisé grâce à la librairie mise à disposition sur GitHub "ChatSharp". Cette librairie sert à se connecter a un chat IRC ainsi que de recevoir les données envoyé sur le chat. La librairie "ChatSharp" offre des classes permettant de créer un utilisateur IRC ainsi qu'un client IRC.

Afin de créer une connexion IRC, il faut un nom d'utilisateur, celui du compte connecté. Un pseudonyme, je mets le nom d'utilisateur. Il faut également un jeton d'accès (cf.4.5.1) qui permet de se connecter aux chats depuis l'extérieur. (cf. figure 5.9)

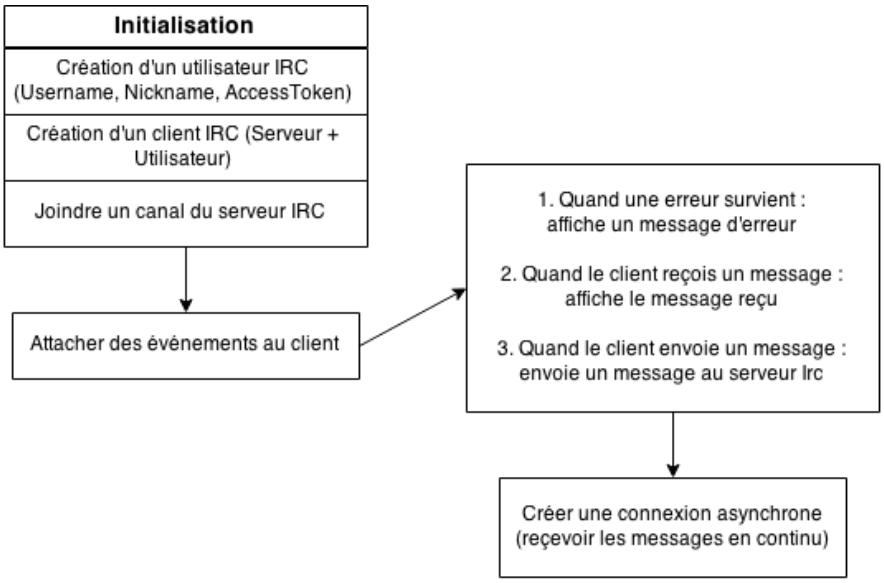


FIGURE 5.9: Étape de connexion à un chat IRC

Les messages reçues des événements sont affiché dans un TextBox.

5.4 Récupération des données

5.4.1 Classes structure

Afin de récupérer les données reçues grâce aux API 4.2, j'ai créé des classes structures. En effet, ces classes ne sont composées que de propriétés et n'ont pas de méthodes. Chaque propriété est une donnée reçue suivant la demande envoyée. (cf. figure 5.10)

Pour spécifier que se sont des classes "structures" il faut spécifier la classe en *[DataContract]* et les propriétés en *[DataMember]*.

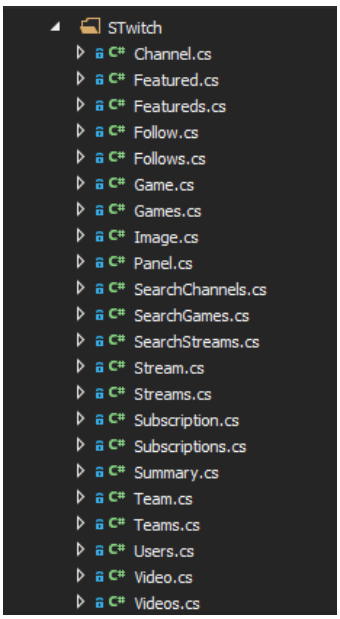


FIGURE 5.10: Classes Twitch

Ci-dessus, les classes "structures" du site Twitch. Il y a une liste de classes pour chaque site intégré à l'application. On peut trouver les différentes classes "structures" dans la documentation des différentes API.

5.4.2 Curl

Client **U**RL **R**equest **L**ibrary est une interface en ligne de commande. Elle permet de récupérer des données d'une ressource accessible par réseau. On désigne la ressource à partir d'une URL. (cf. figure 5.11)

```
curl -H 'Accept: application/vnd.twitchtv.v3+json' -H 'Authorization: OAuth <access_token>' \
-X GET https://api.twitch.tv/kraken/channel
```

FIGURE 5.11: Exemple de commande cURL

Ci-dessus, un exemple de récupération de données sur le site Twitch. La ressource désignée est "https ://api.twitch.tv/kraken/channel".

Il y a plusieurs types de requêtes :
- GET
- POST
- PUT
- DELETE

En C#, afin de récupérer les données d'une ressource, j'utilise la classe proposée par le Framework .NET 4.5 **httpWebRequest**.
J'ai créé une fonction qui permet de récupérer la réponse d'une requête HTML :

```
1             /// <summary>
2             /// Send request and get response html
3             /// </summary>
4             /// <param name="urlRequest">request url</param>
5             /// <param name="p_method">method to use</param>
6             /// <param name="p_access_token">user access token</param>
7             /// <param name="acceptHeader">the accept header html</param>
8             /// <returns>HttpWebResponse</returns>
9             public static Stream SendRequest(string urlRequest, string ↩
                  p_method, string p_access_token, string acceptHeader)
10            {
11              //Create a new http request
12              var httpWebRequest = ↩
                    (HttpWebRequest)WebRequest.Create(urlRequest);
13
14              //Init the http request
15              httpWebRequest.ContentType = "application/json";
16              httpWebRequest.Accept = acceptHeader;
17              httpWebRequest.Method = p_method;
18              httpWebRequest.Headers.Add("Authorization: OAuth " + ↩
                    p_access_token);
19
20              if (p_method == "PUT")
21                httpWebRequest.ContentLength = 0;
22
23              //Create a http response
24              Stream httpResponse = ↩
                    httpWebRequest.GetResponse().GetResponseStream();
25
26              return httpResponse;
27            }
```

Il faut ensuite désérialiser cette réponse. Afin d'avoir une méthode qui renvoie le bon type suivant la ressource demandée, j'ai utilisé le type générique de C#. C'est lors de l'appel de la méthode que l'on spécifie de quel type est la valeur de retour.

```
1     /// <summary>
2     /// Deserialize a http web response
3     /// </summary>
4     /// <typeparam name="T">Generic type</typeparam>
5     /// <param name="jsonContent">content json</param>
6     /// <returns>the object deserialized</returns>
7     public static T Deserialize<T>(Stream jsonContent)
8     {
9       var httpResponse = jsonContent;
10
11      //Read the response
12      using (var streamReader = new StreamReader(httpResponse))
13      {
14        //Add to the generics variable the result
15        T answer = ↩
              JsonConvert.DeserializeObject<T>(streamReader.ReadToEnd());
16        return answer;
17      }
18    }
```

5.5 Authentification

L'authentification se fait grâce au protocole OAuth2 (cf. 4.5.1). Pour rendre l'authentification générique, il faut utiliser le même type d'autorisation. Dans notre cas, le problème est que pas tous les sites utilisent le type d'autorisation implicite.
C'est pour cela que j'ai créé une classe permettant la connexion implicite ainsi que la connexion par autorisation de flux.

5.5.1 Générique

Je rencontre un problème pour la connexion générique, en effet, afin de pouvoir utiliser le protocole OAuth2, il faut que j'utilise ASP.NET [2]. Afin d'éviter l'utilisation d'ASP.NET, je n'utilise pas moi-même le protocole OAuth2 mais je le sous-traite aux API. (cf. 4.2) (cf. figure 5.12)

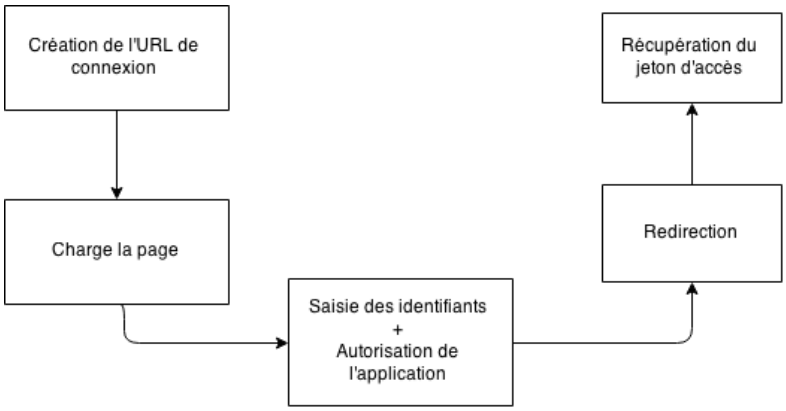


FIGURE 5.12: Authentification

La création de l'URL de connexion se fait par rapport au site sur lequel l'utilisateur veut se connecter. En effet, les services proposent une page Web permettant à un utilisateur de se connecter à son compte sur une application ainsi que de l'autoriser. L'url de cette page Web est composée comme suit :

https ://[url de l'api] ?response_ type=token&client_ id="[id de l'application]"& redirect_ uri="[page de redirection]"&scope="[Liste des droits]"

Le client_id est l'id que donne le service chez qui j'ai enregistré mon application, il varie pour chaque site. La page de redirection sert uniquement à récupérer le jeton d'accès dans le lien, en effet si l'utilisateur saisi correctement ses identifiants, il sera redirigé sur cette page. Il doit également accepter l'utilisation de son compte sur mon application.

2. ensemble de technologies de programmation Web

5.5.2 Connexion automatique

La connexion automatique est la connexion aux différents comptes de l'utilisateur lors de l'ouverture de l'application. Malheureusement, c'est techniquement impossible, en effet, le fait de devoir passer de toute façon par une page web fais que l'on ne peut pas remplir automatiquement les champs d'identifiants.

Le navigateur intégré dans Visual Studio utilise Internet Explorer, et donc prends en compte les cookies. Cela ne remplace pas la connexion automatique, mais l'utilisateur passe moins de temps à se connecter, car les identifiants sont déjà enregistrés, il suffit juste de cliquer sur un bouton pour valider la connexion.

5.5.3 Page web de connexion et d'autorisation

Lorsqu'un utilisateur veut se connecter et ainsi autoriser un compte à utiliser mon application, celui-ci est redirigé vers un site internet. Ce site internet que l'on a accès via : `http://froelicher.github.io/WebMediaManager/`(Page vide) a été mis en ligne par moi-même.

C'est également sur ce site que l'on récupère le token dans l'URL. cf. 5.12

Pour la mise en ligne du site, je me suis servi du service proposé par GitHub : GitHub Pages. Ce service permet d'héberger un site Web sur un dépôt Github et automatiquement, votre site web est en ligne.

5.6 Notification

Afin d'être notifié lorsqu'une vidéo sort ou lorsque une diffusion de flux vidéo en direct commence, je compare deux listes. La première contient les vidéos actuellement sorties et les diffusions en direct, la deuxième va récupérer depuis une requête la liste des vidéos sorties.
Chaque X secondes, on compare les deux listes pour voir s'il y a eu une vidéo qui est en plus dans la deuxième liste. (cf. figure 5.13)

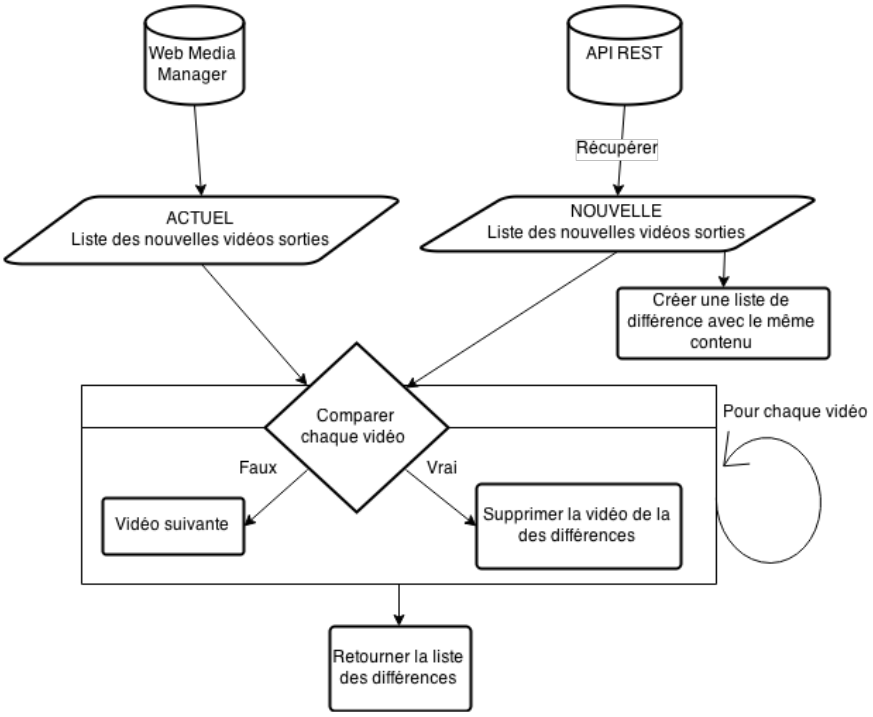


FIGURE 5.13: Notification

J'utilise cette méthode pour détecter les nouvelles vidéos ainsi que les nouvelles diffusions de flux vidéo en direct.

Afin d'avoir une détection automatique qui ne ralentit pas la vue lors de la vérification, j'appelle cette méthode depuis un processus différent de celui de la vue.

```
1     List<List<StreamingSite.SVideo>> newVideos = ↩
          this.SitesController.CheckNotificationsOnlineStreams();
2
3     for (int i = 0; i < newVideos.Count; i++)
4     {
5       this.Invoke((MethodInvoker)delegate
6       {
7         if (newVideos[i].Count > 0)
8         {
9           //Affichage de la notification
10        }
11      }
```

```
12          );
13      }
```

5.7 Affichage

5.7.1 Flux vidéo

Les flux vidéos sont affiché grâce au composant de base proposé par .NET : WebBrowser. En effet, tous les sites proposent un lien afin d'avoir uniquement le lecteur vidéo. J'ai choisi cette, solution car chaque lecteur est différent et ont donc certaine fonctionnalités le sont aussi.

5.7.2 Chat IRC

Comme dis plus haut (cf. 5.3.1), les messages du chat sont affichés dans un TextBox. Afin de pouvoir y afficher depuis un événement, j'ai dû créer un autre processus.

5.7.3 Notification

Le composant externe "PopopNotifier" a été utilisé pour afficher un pop-up lorsqu'une nouvelle diffusion commence. (cf. figure 5.14) Ce composant vient du web : `http://www.codeproject.com/Articles/277584/Notification-Window`.

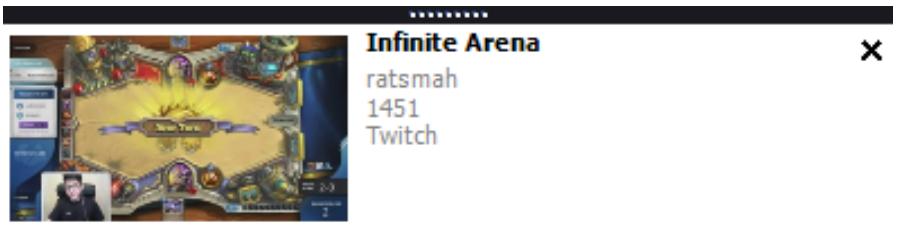


FIGURE 5.14: Exemple de notification

Pour rendre le composant plus attrayant et adapté à mon interface homme-machine, j'ai modifié certains éléments graphiques à l'interne du composant. Les différentes informations affichées : nom de la vidéo ; nom de la chaîne ; nombre de vues ; description de la vidéo

Lorsque c'est une notification de nouvelle diffusion de flux vidéo en direct la description n'est pas affichée.

Chapitre 6

Tests

6.1 Tests fonctionnel

Cette série de tests a été effectuée lorsque l'utilisateur est connecté à au moins un compte. (cf. figure 6.1)

Requiert d'être connecté			
Action testée	**Réalisation**	**Résultat attendu**	**Résultat obtenu**
Lister les stream en ligne	Connexion à un compte quelconque	Redirection sur la vue d'un site avec la liste des streams en ligne	Redirection sur la vue d'un site avec la liste des streams en ligne
Lister les chaînes suivis	Connexion à un compte quelconque	Redirection sur la vue d'un site avec la liste des chaînes suivis	Redirection sur la vue d'un site avec la liste des chaînes suivis
Suivre une chaîne	Rechercher « test » Cliquer sur le premier résultat Cliquer sur le bouton « Abonner »	La chaîne s'affiche la liste des chaînes suivis	La chaîne s'affiche la liste des chaînes suivis
Notification	Suivre une chaîne qui à un stream actuellement en ligne	Une notification du nouveau stream apparaît	Une notification du nouveau stream apparaît
Afficher les détails d'une chaîne	Lister les chaînes suivis Cliquer sur une des chaînes	Vue du de la chaîne	none
Affichage du chat d'un stream	Aller sur la vue d'un stream en ligne	Les messages du chat s'affiche dans le textbox de droite	Les messages du chat s'affiche dans le textbox de droite
Déconnexion	Aller sur la vue d'un site Cliquer sur le bouton « Disconnect »	Redirection sur la vue principale et étant déconnecté	Reste connecté au site

FIGURE 6.1: Tests en étant connecté

Cette série de tests a été effectuée lorsque l'utilisateur n'est connecté à aucun compte. (cf. figure 6.2)

Aucune condition			
Action testée	**Réalisation**	**Résultat attendu**	**Résultat obtenu**
Connexion à un compte Twitch	Ouvrir la fenêtre de connexion Rentrer des identifiants valides	Connecté à un compte Twitch	Connecté à un compte Twitch
Recherche de streams / videos	Rechercher « test »	Affichage du résultat de la recherche	Affichage du résultat de la recherche
Lister les catégories	Cliquer sur « Personal interface »	La liste des catégories s'affiche dans le menu de gauche	La liste des catégories s'affiche dans le menu de gauche
Lister les listes de lectures	Cliquer sur « Personal interface »	La liste des listes de lectures s'affiche dans le menu de gauche	La liste des listes de lectures s'affiche dans le menu de gauche
Ajouter une catégorie	Saisir un nom de catégorie Cliquer sur « Add »	La catégorie s'ajoute dans la liste	La catégorie s'ajoute dans la liste
Ajouter une liste de lectures	Saisir un nom de liste de lectures Cliquer sur « Add »	La liste de lecture s'ajoute dans la liste	La liste de lecture s'ajoute dans la liste
Afficher le contenu d'une catégorie / liste de lectures	Cliquer sur une catégorie ou liste de lectures	Affiche les miniatures des vidéos	Affiche les miniatures des vidéos
Ajouter une vidéo à une catégorie / liste de lectures	Rechercher « test » Cliquer sur la première vidéo Choisir dans la liste des catégories une catégories déjà existante Cliquer sur le bouton « + »	La vidéo est ajoutée dans la catégorie	La vidéo est ajoutée dans la catégorie
Affichage du chat d'un stream	Aller sur la vue d'un stream en ligne	Aucun affichage	Aucun affichage
Affichage d'une vidéo	Rechercher « test » Cliquer sur la première vidéo	Affiche la vue d'une vidéo	Affiche la vue d'une vidéo

FIGURE 6.2: Tests en étant déconnecté

Chapitre 7

Conclusions

7.1 Apport personnel

L'estimation de l'apport personnel apporté sur ce projet est : La création de la vue est 80% de moi-même, l'affiche du lecteur vidéo est fait grâce au composant WebBrowser. La vue de notification est un composant externe où les propriétés graphiques ont été modifiées par moi.
La méthode cURL a été faite 100% par moi.
Le chat IRC a été prit sur le Web, mais mis en place 100% par moi-même.
L'authentification est faite à 50% par moi, j'affiche la page de connexion et récupère les informations, le serveur lui s'occupe de me donner le jeton d'accès.

7.2 Conclusion personnel

J'ai eu plaisir de travailler sur ce projet, en effet le thème des vidéos est un thème que j'affectionne tout particulièrement. C'est une application que moi-même peut utiliser, mais pas que, beaucoup de mes amis également.
La gestion du temps a encore posé problème, bien que le travail a été effectué tout au long du diplôme, je n'ai pas su me concentrer au bon moment afin de finir tranquillement.
Le plus long et dur pour moi est la conception de l'application, en effet, je passe beaucoup de temps à concevoir l'application par rapport à la réalisation. Malgré tout, je suis tout de même satisfait de l'application réalisée même si elle ne remplit pas entièrement le cahier des charges.

7.3 Conclusion technique

7.3.1 Bilan de réalisation

Au niveau de la réalisation, l'application ne remplit malheureusement pas le cahier des charges dans son intégralité. Je pense avoir mis trop de fonctionnalités, même si elles

ne sont pas dure à réaliser. Toutes les fonctionnalités qui ne sont pas intégrées sont celles que l'utilisateur utilise le moins. L'application est tout de même fonctionnelle.

7.3.2 Améliorations et perspectives

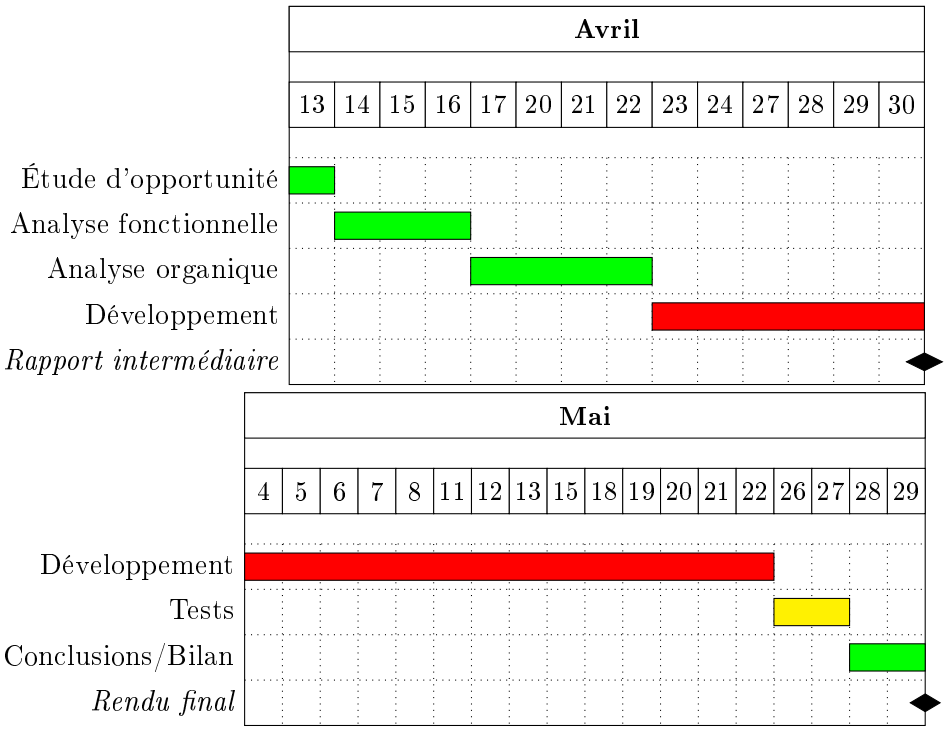
Tout d'abord finir l'application avec toutes les fonctionnalités prévues initialement. Puis, ajouter du contenu, c'est-à-dire intégrer d'autre site vidéos à mon application, étant donné qu'elle est conçue pour l'évolution justement. Afin de rendre l'utilisation plus agréable, faire une meilleure interface graphique ainsi qu'optimiser l'application. Afin d'avoir une connexion complètement à l'interne, utiliser ASP.NET.

Chapitre 8

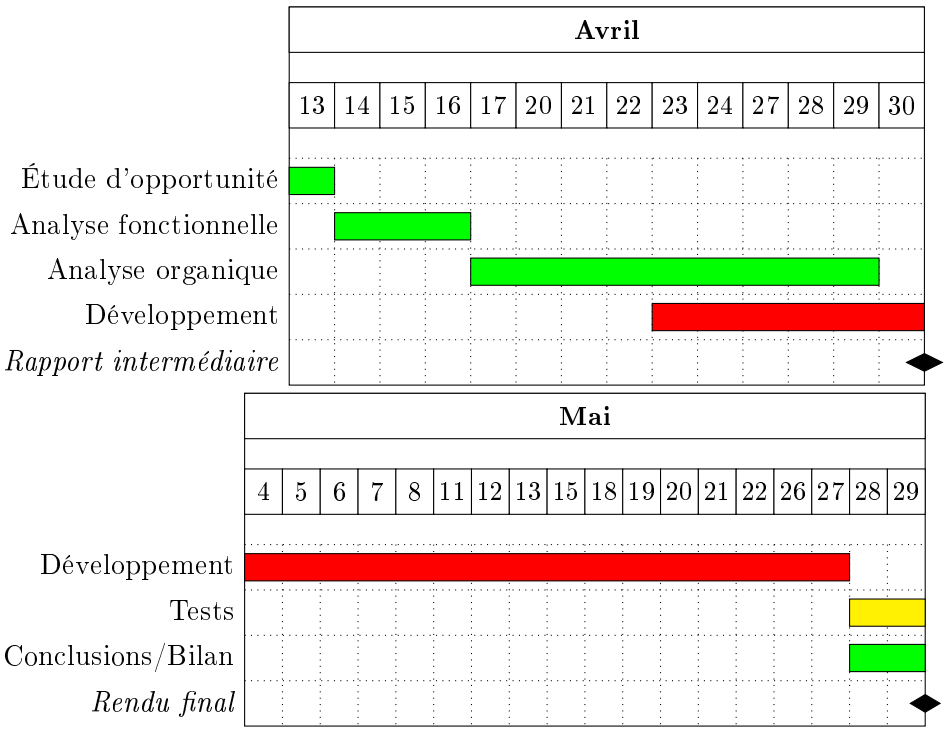
Annexes

8.1 Planning

8.1.1 Initial



8.1.2 Final



8.1.3 Sources

Documentation API Twitch , *Github Twitch*, `https://github.com/justintv/Twitch-API`, Mai 2015
Documentation API Youtube, *Api Youtube*, `https://developers.google.com/youtube/v3/docs/`, Mai 2015
Documentation API Vimeo, *Api Dailymotion*, `https://developer.dailymotion.com/documentation`, Mai 2015
Documentation API Dailymotion, *Api Vimeo*, `https://developer.vimeo.com/api`, Mai 2015
Source IRC chat, *GitHub ChatSharp*, `https://github.com/SirCmpwn/ChatSharp`, Mai 2015
Code Project NotificationWindows, *Source project*,`http://www.codeproject.com/Articles/277584/Notification-Window`, Mai 2015
WebMediaManager GitHub, *GitHub project*, `https://github.com/Froelicher/WebMediaManager`, Mai 2015

Table des figures