

Web Media Manager

The Author

27 mai 2015

Table des matières

1	Résumés	4
2	Introduction	5
3	Cahier des charges	6
3.1	Titre du projet	6
3.2	Objectifs du projet	6
3.3	Description détaillée	6
3.4	Inventaire des étapes	7
3.5	Inventaire du matériel	7
3.6	Inventaire des logiciels	7
3.7	Délivrables (documents à restituer)	7
3.8	Éléments mesurables (servant à l'évaluation)	8
4	Étude d'opportunité	9
4.1	Introduction au projet	10
4.1.1	Médias vidéo web	10
4.1.2	Pourquoi avoir choisi ce sujet ?	10
4.2	Analyse de l'existant	11
4.2.1	Existant	11
4.2.2	Critique de l'existant	11
5	Analyse fonctionnelle	12
5.1	Description du fonctionnement	12
5.1.1	Fonctions des sites	12
5.1.2	Gestion des catégories	13
5.1.3	Gestion des listes de lecture	13
5.1.4	Notifications	13
5.2	API	13
5.2.1	API REST	13
5.3	Lecteur vidéo	14
5.3.1	Flash player	14
5.3.2	HTML5	15

5.4	Outil communautaire	15
5.4.1	Chat IRC	15
5.4.2	Commentaires	15
5.5	Connexion	16
5.5.1	OAuth2	16
5.6	Interface homme-machine	18
5.6.1	Miniature	18
5.6.2	Notification	18
5.6.3	Navigation	18
5.6.4	Interface personnelle	19
5.6.5	Interface des sites	20
5.6.6	Interface vidéo	21
5.6.7	Interface détail d'un compte	22
5.6.8	Interface de connexion	22
6	Analyse organique	23
6.1	Généralité	23
6.1.1	Langage en environnement de développement	23
6.1.2	Format des fichiers et enregistrements	23
6.1.3	SVideo & SChannel	23
6.2	Conception	25
6.2.1	Modèle	25
6.2.2	Catégorie et liste de lectures	25
6.2.3	Global	25
6.3	Outil communautaire	26
6.3.1	IRC	26
6.4	Récupération des données	27
6.4.1	Classes structure	27
6.4.2	Curl	27
6.5	Authentification	31
6.5.1	Générique	31
6.5.2	Connexion automatique	31
6.6	Notification	33
6.7	Affichage	34
6.7.1	Flux vidéo	34
6.7.2	Notification	34
7	Tests	35
7.1	Tests fonctionnel	35
7.2	Tests unitaire	35
8	Conclusions	36

9	Annexes	37
9.1	Planning	37
9.1.1	Initial	37
9.1.2	Final	37

Chapitre 1

Résumés

Chapitre 2

Introduction

Chapitre 3

Cahier des charges

3.1 Titre du projet

WebMedia Manager

3.2 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 au 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.

3.3 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. propose des services de bases pour leurs utilisateurs. Elle reprend si possible dynamiquement les différentes fonctions de chaque site et les propose dans une interface crée 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 sites à quelques exception près. Ci-dessous, les fonctions ajoutée 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#.

3.4 Inventaire des étapes

Début : Lundi 13/04/2015

Reddition intermédiaire (doc + poster) : Vendredi 30/04/2015

Reddition finale : Lundi 01/06/2015

3.5 Inventaire du matériel

PC + 2 écrans

3.6 Inventaire des logiciels

Visual Studio 2013 Professionnel

3.7 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

3.8 Éléments mesurables (servant à l'évaluation)

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

Chapitre 4

Étude d'opportunité

4.1 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.

4.1.1 Médias vidéo web

Aujourd'hui, les sites comme Youtube, Dailymotion, Twitch qui proposent 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 certains "créateurs de contenu audiovisuel" sont même payés par ces sites par rapport à leur popularité. De plus en plus de personnes, surtout les jeunes, passent leurs temps devant des vidéos sur le web que sur 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 c'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 plus part 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 leur communauté, certains font ça à plein temps et donc sont contraints d'entretenir une grande communauté.

Différé

Les vidéos "différées" sont depuis un petit moment déjà présentes 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 a pris 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.

4.1.2 Pourquoi avoir choisi ce sujet ?

J'ai choisi ce sujet car j'ai porté un réel intérêt au monde audiovisuel sur le web. En effet, je fais parti des jeunes qui ont 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 utilisent ces sites régulièrement. Le fait que le monde de l'audiovisuel est en plein boom 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 site proposant à ses utilisateurs de pouvoir diffuser du flux vidéo en direct, ce qui consolide l'intérêt de mon application.

4.2 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 présente sur les différents sites.

4.2.1 Existant

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

4.2.2 Critique de l'existant

Twitch now : Cette application permet à l'utilisateur de naviguer dans les différentes diffusion 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 diffusion 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 diffusion sont classé soit par vues, soit par jeux. Évidemment, les diffusions de flux vidéo peuvent être visionnée depuis l'application. Dailymotion Games est une application officiel de Dailymotion.

Chapitre 5

Analyse fonctionnelle

5.1 Description du fonctionnement

5.1.1 Fonctions des sites

Générique

Tout les sites proposent des fonctions "commune" de base qui sont intégrée 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 spécifique aux sites qui proposes des diffusions en direct :

- *Chat* : Discuter avec les autres spectateurs
- Abonnement payant

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

5.1.2 Gestion des catégories

Les catégories servent à rassembler plusieurs vidéos de différents sites. En effet, grâce aux catégories, les vidéos peuvent être organisées 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.

5.1.3 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 possible sur une vidéo :

- Supprimer
- Monter (dans la liste)
- Descendre (dans la liste)
- Déplacer à X position (dans la liste)

5.1.4 Notifications

Les notifications servent à être tenu 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.

5.2 API

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

5.2.1 API REST

Les API¹ REST² servent à accéder à une ressource par son URI grâce au protocole HTML pour procéder à diverses opérations : GET ; POST ; PUT ; DELETE.

1. Application Programming Interface

2. Representational State Transfer

Le format de représentation des données est libre, dans le cas des API que l'on utilise, le format est en JSON³.

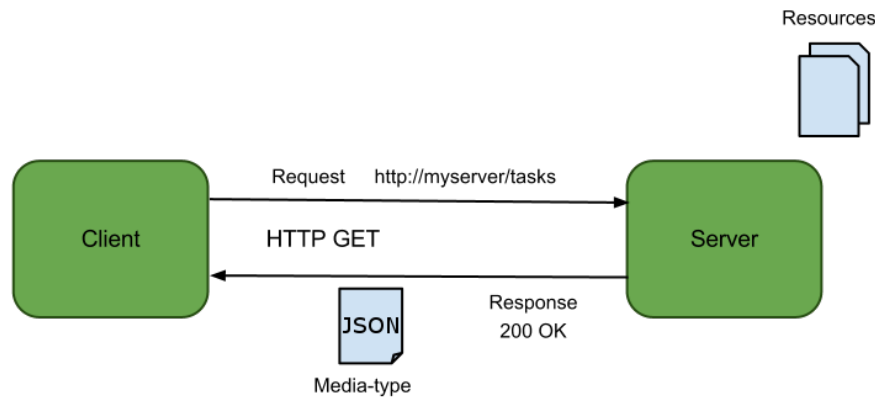


FIGURE 5.1: REST

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.

5.3 Lecteur vidéo

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

5.3.1 Flash player

Le plus part des sites utilisent le lecteur flash pour leurs lecteurs vidéo. Le lecteur flash utilise le protocole RTMP⁴, 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.

3. JavaScript Object Notation

4. Real Time Messaging Protocol

5.3.2 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'accès au contenu HD 4k à 60 fps⁵. Le fait d'utiliser le lecteur HTML5 est un plus dans le multi-plateforme.

5.4 Outil communautaire

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.

5.4.1 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 plus part des *chat* utilisé pour les diffusions de flux vidéo en direct sont des *chats* IRC⁶.

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 de un à un. Le principe est que chaque utilisateur est connecté à un serveur IRC qui eux-même sont relié avec d'autres serveurs IRC. Ainsi toutes les personnes peuvent discuter sur des forums publics ou privé.

Les différents site de diffusion 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.

5.4.2 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.

5. Frame Per Second

6. Internet Relay Chat

5.5 Connexion

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

5.5.1 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.

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. Un token 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(*Authorization 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

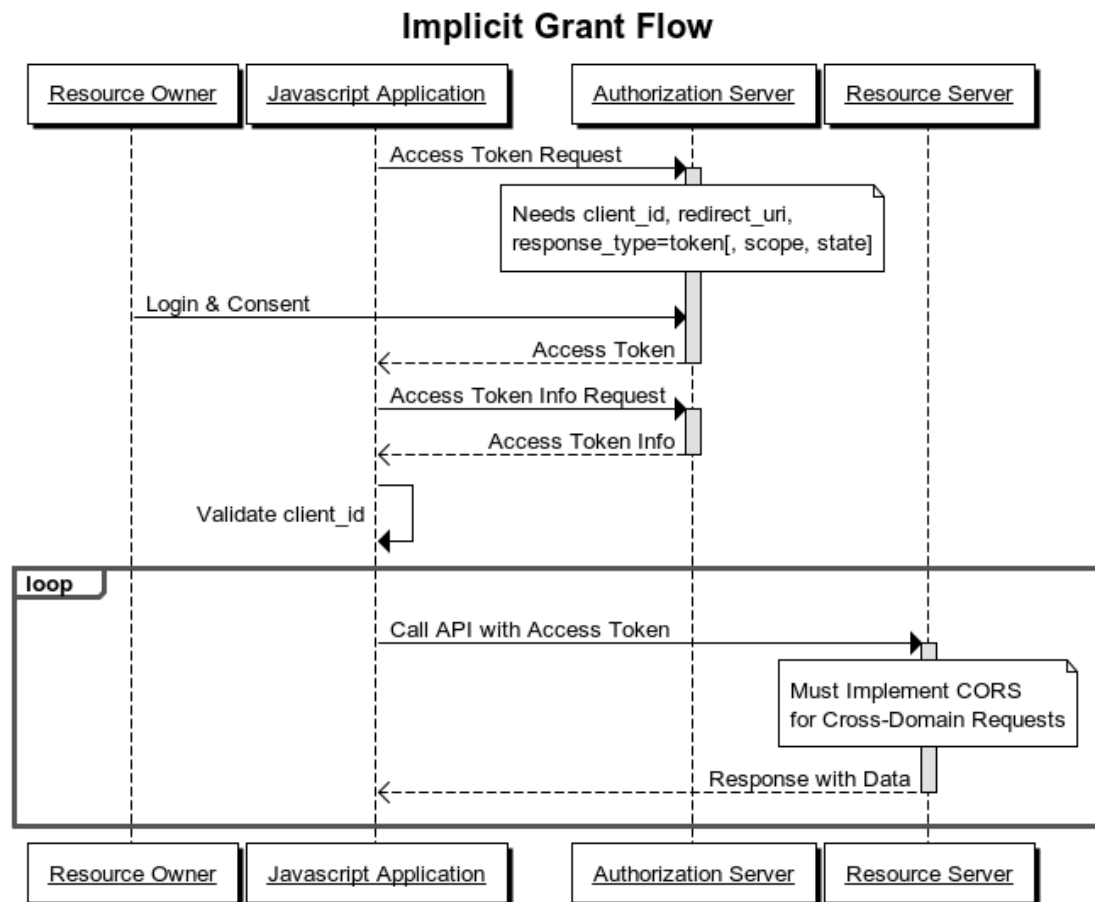


FIGURE 5.2: Schéma OAuth2 : Droit implicite

5.6 Interface homme-machine

5.6.1 Miniature

Les miniatures des vidéos sont affichées sous cette forme

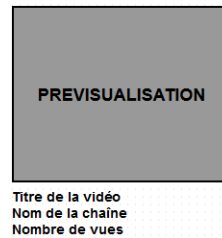


FIGURE 5.3: Miniature

5.6.2 Notification

Les notifications sont affichées sous cette forme

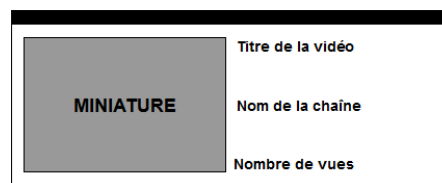


FIGURE 5.4: Interface notification

5.6.3 Navigation

Les couleurs représentent les différentes fenêtres de l'application.

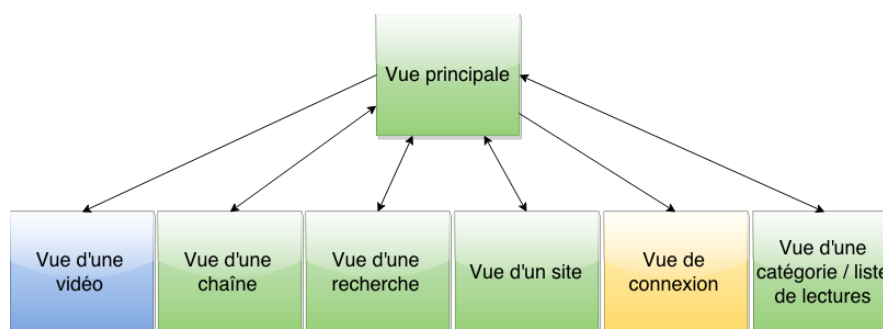


FIGURE 5.5: Navigation de l'application

5.6.4 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.



FIGURE 5.6: Interface personnel

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

5.6.5 Interface des sites

L'interface des sites est commune pour tous les sites. Chaque site à une interface générique.

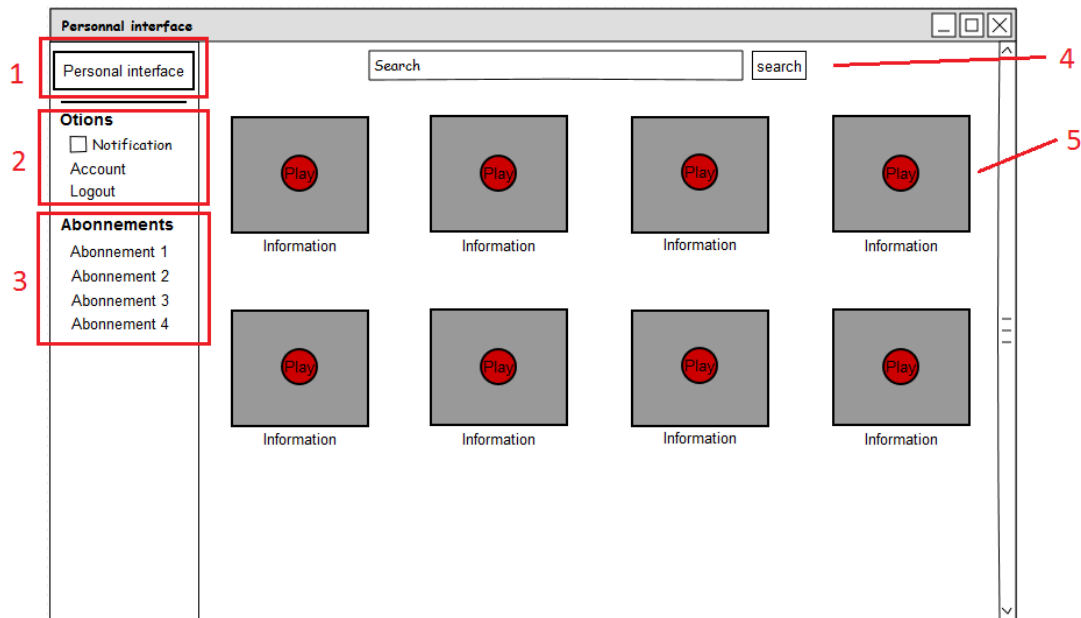


FIGURE 5.7: Interface des sites

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

5.6.6 Interface vidéo

L'interface vidéo est générique, pour chaque vidéo de chaque site c'est la même interface.



FIGURE 5.8: Interface d'une vidéo

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* ou espace commentaires
7. Nombre de vues de la vidéo

5.6.7 Interface détail d'un compte

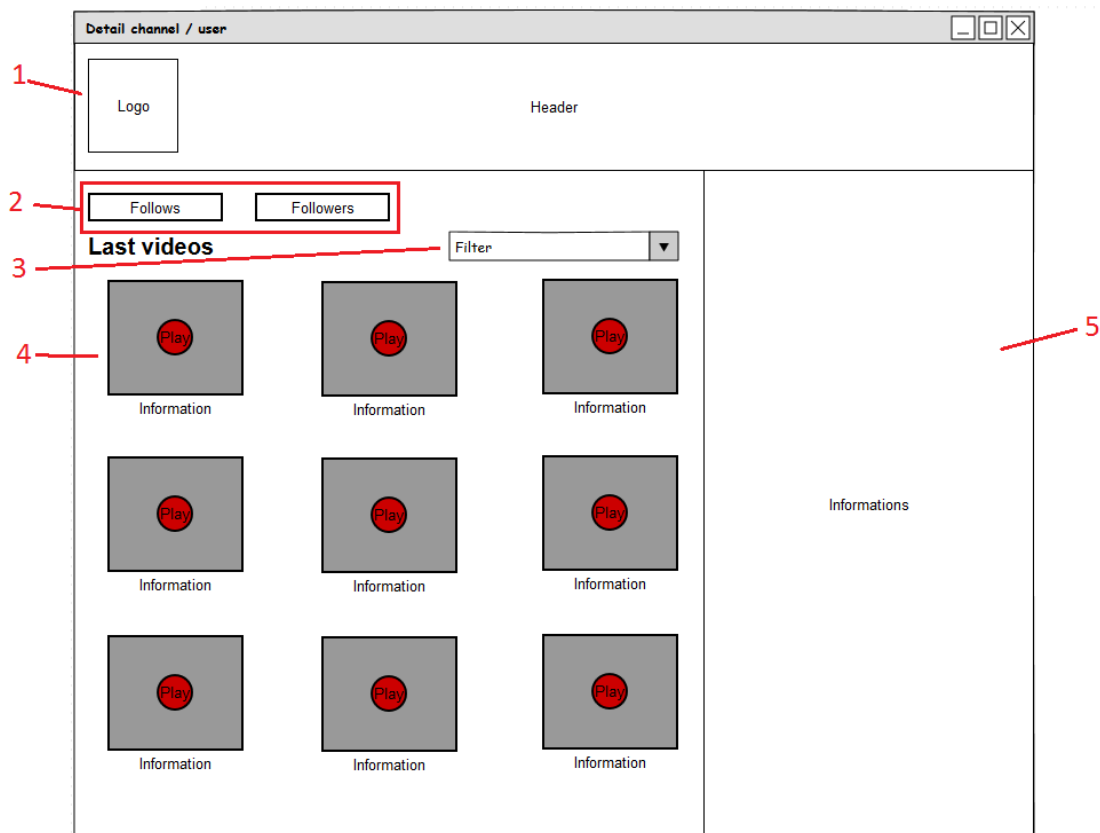


FIGURE 5.9: Interface d'une chaîne / utilisateur

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

5.6.8 Interface de connexion

Chapitre 6

Analyse organique

6.1 Généralité

6.1.1 Langage en environnement de développement

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

6.1.2 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 contenu. Elle à également besoin d'enregistrer quelques configurations. C'est pour ça que j'utilise le format de fichier INI¹ car il est très facile à manipuler.

```
[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 6.1: Exemple de fichier INI

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

6.1.3 SVideo & SChannel

J'ai crée des formats génériques pour les vidéos des différents services. En effet, c'est pour pouvoir traiter toutes les vidéos de la même manière et ainsi pouvoir par exemple afficher toutes les miniatures grâce à la même méthode.

SVideo : Structures pour les vidéos et vidéos en direct

1. Fichier de configuration


```
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 6.2: Structure SVideo

SChannel : Structure pour les chaînes

```
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 6.3: Structure SChannel

6.2 Conception

6.2.1 Modèle

StreamingSite

Pour pouvoir traiter les différents sites de la même manière, j'ai créé 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.

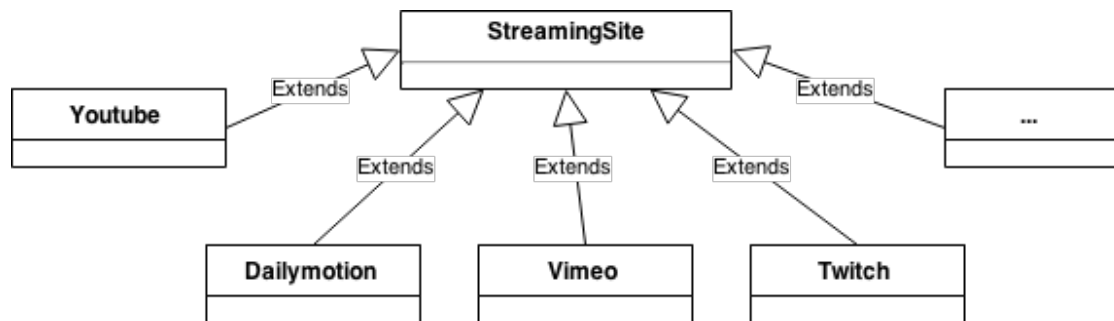


FIGURE 6.4: Diagramme StreamingSite

Pour ajouter d'autres sites il faut juste créer une classe qui respecte la classe de base.
TODO : Mettre une table de StreamingSite

6.2.2 Catégorie et liste de lectures

Pour différencier les catégories et les liste de lectures j'ai créé une classe de base *Container* qui contient toute les méthodes concernant les deux types de conteneur. Étant donné que les liste 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.

6.2.3 Global

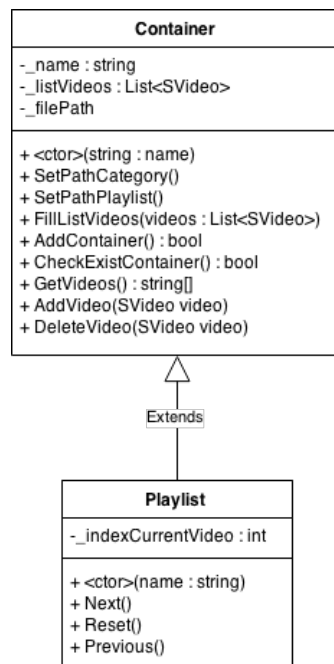


FIGURE 6.5: Diagramme Container

6.3 Outil communautaire

6.3.1 IRC

Le chat IRC a été réalisé grâce à la librairie mise à disposition sur GitHub "Chat-Sharp". Cette librairie sert à se connecter à un chat IRC ainsi que de recevoir les données envoyées sur le chat.

La librairie "ChatSharp" offre des classes permettant de créer un utilisateur IRC ainsi qu'un client IRC.

Ci-dessous les étapes d'une connexion à un chat IRC

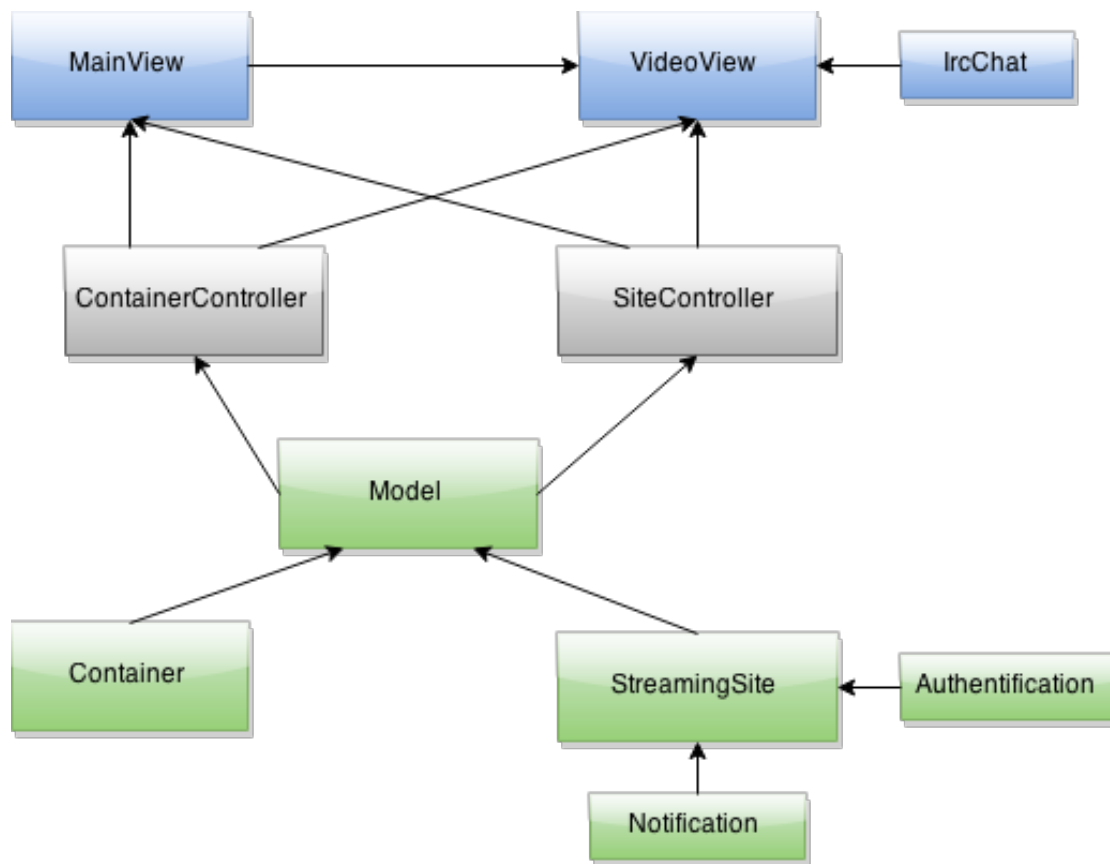


FIGURE 6.6: Diagramme global

6.4 Récupération des données

6.4.1 Classes structure

Afin de récupérer les données reçues grâce aux API, 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.

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

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.

6.4.2 Curl

Client URL Request Library est une interface en ligne de commande. Elle permet de récupérer des données d'une ressource accessible par réseau. On désigne la ressource à

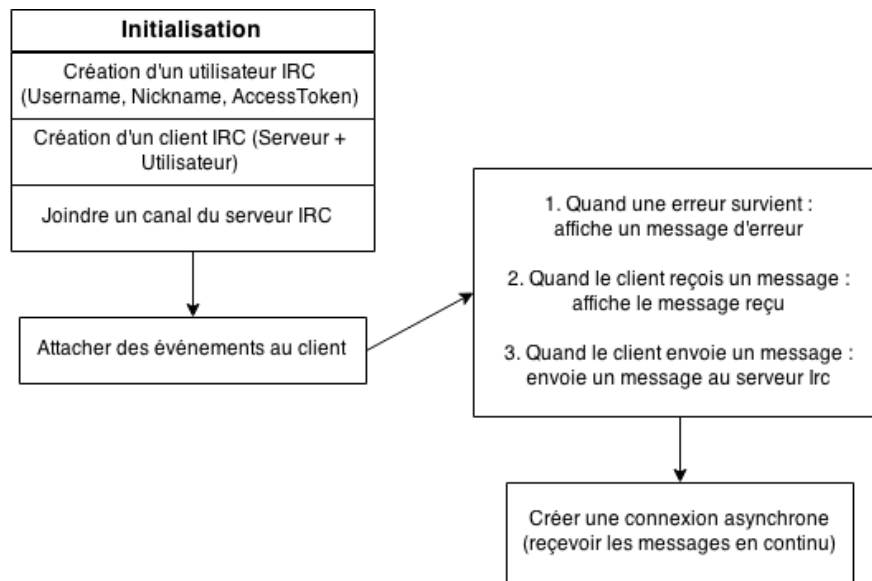


FIGURE 6.7: Étape de connexion à un chat IRC

partir d'une URL.

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 type de requêtes :

- GET
- POST
- PUT
- DELETE

En C#, afin de récupérer les données d'une ressource, j'utilise la classe fournie par le Framework .NET 4.5 **HttpRequest**.

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 httpRequest = ↵
            (HttpRequest)WebRequest.Create(urlRequest);
13
14         //Init the http request
15         httpRequest.ContentType = "application/json";
  
```

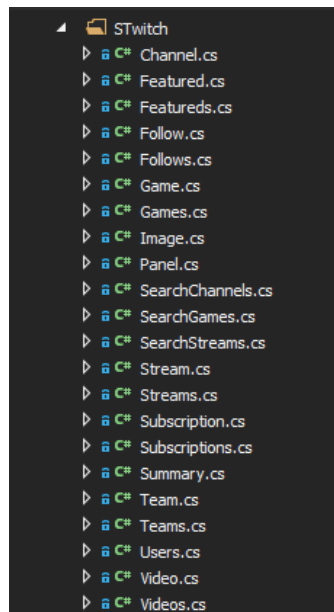


FIGURE 6.8: Classes Twitch

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

FIGURE 6.9: Exemple de commande cURL

```
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 = ↵
16                 JsonConvert.DeserializeObject<T>(streamReader.ReadToEnd());
17             return answer;
18         }
19     }
```

6.5 Authentification

L'authentification se fait grâce au protocole OAuth2[AJOUTER REFERENCE]. Pour rendre l'authentification générique il faut utiliser le même type d'autorisation. Dans notre cas, le problème est que pas tout 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.

6.5.1 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. Afin d'éviter l'utilisation d'ASP.NET, je n'utilise pas moi même le protocole OAuth2 mais je le sous-traire aux API. Je procède de cette manière :

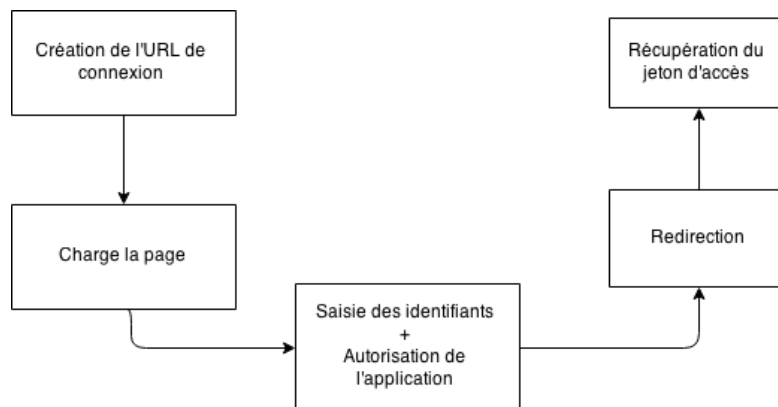


FIGURE 6.10: Authentification

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.

TODO : EXPLIQUER EN DETAILS LA CREATION DURL + IMAGE ?

Une fois les identifiants saisis et l'autorisation d'utilisation du compte sur cette application acceptée, la page web nous redirige vers un lien de redirection que nous avons spécifié auparavant. Dans ce lien, se trouve en paramètre le jeton d'accès délivré par le serveur d'autorisation.

6.5.2 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 fait 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 enregistré, il suffit juste de cliquer sur un bouton pour valider la connexion.

6.6 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 si il y a eu une vidéo qui est en plus dans la deuxième liste.

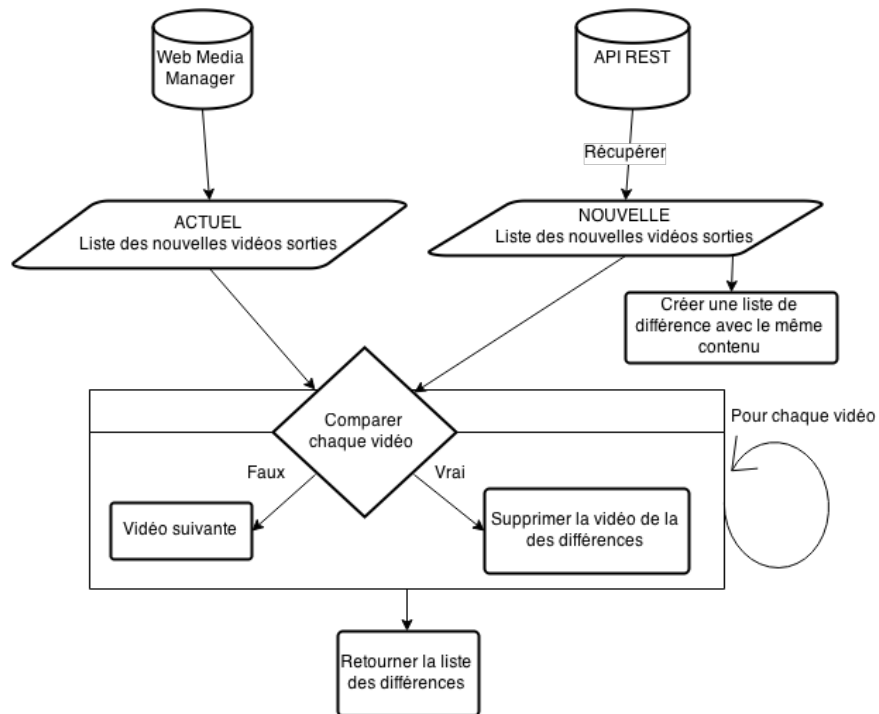


FIGURE 6.11: Notification

6.7 Affichage

6.7.1 Flux vidéo

Les flux vidéos sont affichés 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 certaines fonctionnalités les sont aussi.

6.7.2 Notification

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

TODO : METTRE IMAGE

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

Chapitre 7

Tests

7.1 Tests fonctionnel

7.2 Tests unitaire

Chapitre 8

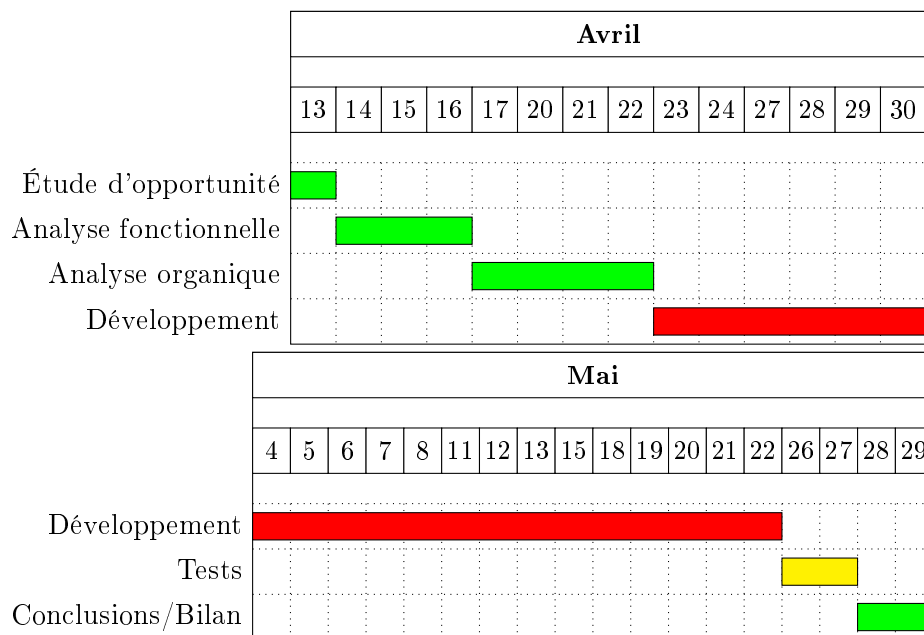
Conclusions

Chapitre 9

Annexes

9.1 Planning

9.1.1 Initial



9.1.2 Final

Table des figures

5.1	REST	14
5.2	Schéma OAuth2 : Droit implicite	17
5.3	Miniature	18
5.4	Interface notification	18
5.5	Navigation de l'application	18
5.6	Interface personnel	19
5.7	Interface des sites	20
5.8	Interface d'une vidéo	21
5.9	Interface d'une chaîne / utilisateur	22
6.1	Exemple de fichier INI	23
6.2	Structure SVideo	24
6.3	Structure SChannel	24
6.4	Diagramme StreamingSite	25
6.5	Diagramme Container	26
6.6	Diagramme global	27
6.7	Étape de connexion à un chat IRC	28
6.8	Classes Twitch	29
6.9	Exemple de commande cURL	29
6.10	Authentification	31
6.11	Notification	33