



# Twitch Alert

Travail de semestre

---

CFPT Informatique  
Jean-Philippe Froelicher  
T.IS-E2A

2014 - 2015

## Table des matières

<b>I</b>	<b>Résumé</b>	<b>4</b>
<b>1</b>	<b>Fr</b>	<b>4</b>
<b>2</b>	<b>En</b>	<b>4</b>
<b>II</b>	<b>Cahier des charges</b>	<b>5</b>
<b>3</b>	<b>Initial</b>	<b>5</b>
3.1	Titre du projet . . . . .	5
3.2	Objectifs du projet . . . . .	5
3.3	Description du projet . . . . .	5
<b>III</b>	<b>Étude d'opportunité</b>	<b>7</b>
<b>4</b>	<b>Introduction au projet</b>	<b>7</b>
4.1	Twitch . . . . .	7
4.1.1	Chaîne . . . . .	7
4.1.2	Team . . . . .	7
4.1.3	Suivis (Following) . . . . .	7
4.2	Pourquoi avoir choisi ce sujet ? . . . . .	7
4.3	Pourquoi une application de bureau ? . . . . .	8
4.4	Pourquoi C# ? . . . . .	8
<b>5</b>	<b>Analyse de l'existant</b>	<b>8</b>
5.1	Existant . . . . .	8
5.2	Critique de l'existant . . . . .	8
<b>IV</b>	<b>Analyse fonctionnelle</b>	<b>9</b>
<b>6</b>	<b>Gestion des chaînes</b>	<b>9</b>
<b>7</b>	<b>Recherche d'une chaîne</b>	<b>9</b>
<b>8</b>	<b>Suivre une chaîne</b>	<b>9</b>
<b>9</b>	<b>Gestion des notifications</b>	<b>9</b>
<b>10</b>	<b>Interface homme-machine</b>	<b>10</b>
10.1	Vue principale . . . . .	10
10.2	Vue d'une notification . . . . .	11
10.3	Vue détail d'une chaîne . . . . .	12
10.3.1	Vue de connexion . . . . .	13
<b>V</b>	<b>Analyse organique</b>	<b>14</b>
<b>11</b>	<b>Diagramme de classe</b>	<b>14</b>
11.1	Model . . . . .	14
11.2	Controllers . . . . .	15
11.2.1	Views . . . . .	16

<b>12 API Twitch</b>	<b>17</b>
12.1 Récupération des données . . . . .	17
12.2 OAuth 2.0 . . . . .	18
12.3 cURL . . . . .	20
<b>13 Site d'autorisation et de connexion</b>	<b>21</b>
<b>14 Affichage</b>	<b>21</b>
14.1 Pop-up de notification . . . . .	21
14.2 Flux vidéo . . . . .	21
<b>15 Notification</b>	<b>21</b>
15.1 Vérification de nouveaux streams en ligne . . . . .	21
15.1.1 Multi-Threading . . . . .	22
<b>VI Tests</b>	<b>23</b>
<b>16 Tests fonctionnels</b>	<b>23</b>
<b>VII Conclusion</b>	<b>24</b>
<b>17 Apport personnel</b>	<b>24</b>
<b>18 Conclusion personnelle</b>	<b>24</b>
18.1 Bilan personnel . . . . .	24
<b>19 Conclusion technique</b>	<b>24</b>
19.1 Bilan de réalisation . . . . .	24
19.2 Améliorations et perspectives . . . . .	24
<b>VIII Annexes</b>	<b>24</b>
19.3 Planification . . . . .	25
19.3.1 Planification prévisionnelle . . . . .	25
19.3.2 Planning réel . . . . .	25
<b>IX Références</b>	<b>25</b>

## Première partie

# Résumé

### 1 Fr

Le but de ce projet est de créer une application windows en C#. Cette application doit notifier l'utilisateur des nouvelles diffusions de flux vidéo en direct de Twitch grâce à un petit pop-up. Elle doit également offrir une gestion des notifications et des "Suivis" Twitch. L'application est composée de trois vues : une vue principale pour la navigation entre les différents streams ; une vue de connexion pour la connexion au compte Twitch ; une vue des détails d'une chaîne Twitch. Le flux vidéo est accessible via la vue des détails d'une chaîne. Afin d'obtenir les informations dont j'avais besoin, j'ai utilisé l'API Twitch qui utilise la librairie cURL pour l'envoi de requêtes HTML. Pour l'aspect graphique de la notification, j'ai utilisé un composant trouvé sur le web nommé "Popup Notifier", je l'ai modifié graphiquement pour qu'il convienne à mon application. Je n'ai malheureusement pas pu faire la gestion de notification mais l'application est tout de même fonctionnelle.

### 2 En

The objectives of this project is to create a Windows application in C#. This application must notify the user of new online streams of Twitch with a little pop-up. This application must too provide a management of notifications and follows Twitch. The application is composed by three views : main view for the navigation between the streams ; login view for the connexion of the Twitch account ; details view of channel. The stream is accessible since then the details view of channel. To obtain the informations whose i needed, i use the Twitch API who use the cURL library for send a HTML request. For the graphic aspect of the notification, i use a componant find in the web named "Popup Notifier", i have modify graphicaly for my application. I have unfortunately can't realise the management notification but the application run.

## Deuxième partie

# Cahier des charges

### 3 Initial

#### 3.1 Titre du projet

*Twitch Alert*

#### 3.2 Objectifs du projet

Créer une application de notification(alerte) pour la plateforme de diffusion de *streaming TWITCH*.

Apprentissage et prise en main de l'API <sup>1</sup> Twitch.

#### 3.3 Description du projet

##### Glossaire :

stream (n.m) : Flux vidéo.

streamer (v.) : L'action de diffuser du flux vidéo.

streaming (n.m) : Diffusion de vidéos en continu.

streameur (n.m) : Personne diffusant une vidéo en continu.

Twitch : Site permettant à n'importe qui de diffuser du flux vidéo, chaque utilisateur a une chaine Twitch permettant de diffuser du flux vidéo.

Chaîne Twitch : Un utilisateur Twitch a forcément une chaine Twitch qu'il utilise que lorsqu'il diffuse du flux vidéo.

##### Général :

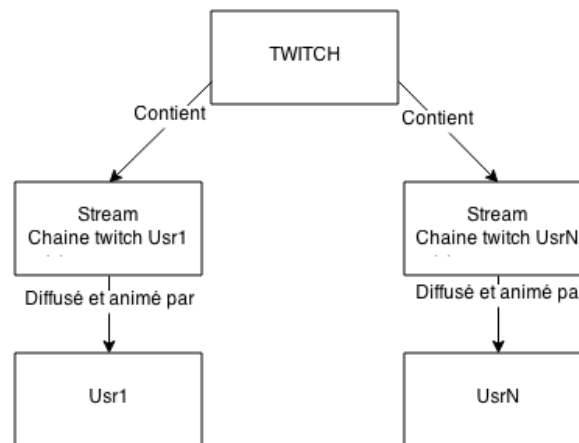
Une vue principale qui permet de configurer les différentes notifications et alertes pour chaque flux vidéo (stream).

L'application tourne en tâche de fond, c'est-à-dire qu'une icône apparait dans la barre de tâche lorsque l'on réduit la fenêtre de l'application. Pour y accéder on clique sur l'icône de la barre de tâche.

Une notification est représentée sous forme de *pop-up*(petite fenêtre) qui apparait en bas à droite de l'écran principal, un petit son est émis lorsque la notification apparait si l'utilisateur ne la voit pas.

---

1. Application Programming Interface



### Twitch

FIGURE 1 – Schema Twitch

- La plateforme Twitch permet de suivre des streameurs "indépendants", actuellement, lorsqu'un streameur commence son stream, une notification est envoyée par e-mail. Le but est de pouvoir saisir le nom Twitch du streameur, et lorsque il commence son stream, mon application notifie l'utilisateur.
- Pouvoir regarder le stream d'un utilisateur Twitch.
- Gestion des notification pour chaque chaine Twitch.
- Gestion des plages horaires de notifications pour chaque chaine Twitch

## Troisième partie

# Étude d'opportunité

## 4 Introduction au projet

Le but de ce projet est de réaliser une application de notification pour la plateforme de diffusion de streaming Twitch ainsi que toute la gestion de ses notifications.

### 4.1 Twitch

Twitch est une plateforme de diffusion de streaming et de VOD<sup>2</sup> de jeux vidéo créée en 2011. C'est la plateforme numéro un pour les amateurs de streaming, toutes personnes voulant commencer dans ce monde est quasiment obligé de passer par Twitch.

La plateforme donne la possibilité de pouvoir transmettre du flux vidéo sur la chaîne Twitch de l'utilisateur en question ainsi, chaque utilisateur peut devenir s'il le souhaite une sorte de mini télévision qui diffuse du flux vidéo lorsqu'il veut.

Les utilisateurs qui diffusent du flux vidéo sont rémunéré grâce aux dons que les spectateurs leur offrent via un service de Twitch.

#### 4.1.1 Chaîne

Une chaîne Twitch offre une multitude de possibilités pour l'utilisateur spectateur, effectivement une chaîne n'est pas simplement un flux vidéo. Il y a la fonction "Suivre" une chaîne Twitch, qui permet aux spectateurs de pouvoir être tenu au courant via e-mail, mais aussi d'accéder plus rapidement à la chaîne en question. Il y a également une fonction "S'abonner" qui est payante, le prix varie selon la chaîne. En revanche, cette fonction offre quelques avantages pas très significatifs, car l'abonnement sert surtout à soutenir financièrement directement la chaîne Twitch. Une chaîne contient également un *chat* permettant à la communauté de cette chaîne de discuter sur le flux vidéo actuel et également avoir une interaction avec l'utilisateur qui diffuse son flux vidéo.

#### 4.1.2 Team

Une *team* Twitch est un groupe de chaîne Twitch, la notion de team permet uniquement de retrouver les différentes chaînes d'un groupe de streaming, ou d'une team professionnelle de jeu etc...

#### 4.1.3 Suivis (Following)

Un utilisateur Twitch a la possibilité de suivre une chaîne Twitch, le principe du "suivis" est que chaque fois que la chaîne commence à diffuser du flux vidéo, les utilisateurs qui suivent cette chaîne sont prévenus via e-mail.

## 4.2 Pourquoi avoir choisi ce sujet ?

J'ai choisi ce sujet parce que je porte un réel intérêt au monde du streaming. En effet, je regarde régulièrement des streams sur des chaînes de la plateforme de streaming Twitch. N'étant jamais au courant de quel streamer est en train de diffuser à moment donné, j'ai donc pensé à une application qui notifie et informe l'utilisateur des différents streams en cours.

De plus, le monde du streaming est tout nouveau ce qui accentue mon intérêt sur le sujet.

---

2. Video On Demand

### 4.3 Pourquoi une application de bureau ?

Au départ, je voulais créer une application mobile, ce qui augmente l'intérêt de l'application, mais le temps laissé à disposition n'est pas assez conséquent et donc étant novice dans la programmation mobile, j'ai décidé de créer une application de bureau que je pourrai peut-être porter plus tard sur mobile.

### 4.4 Pourquoi C# ?

J'ai choisi de développer mon application en C#, car c'est le langage que je connais le mieux actuellement, et je pense également que c'est le langage qui est le plus adapté pour l'application à réaliser, car elle est sur Windows.

## 5 Analyse de l'existant

### 5.1 Existant

Étant donné que le streaming est un monde nouveau il y a peu d'application sur ce sujet, en effet les seules applications sont :

- **Twitch** : Application mobile officielle de la plateforme de streaming Twitch
- **Twitch now** : Application Google Chrome
- **Twitch Notifier** : Application Google Chrome

### 5.2 Critique de l'existant

**Twitch** : Cette application sert à accéder à la plateforme de streaming Twitch. En effet, c'est l'application officielle, il y a donc la possibilité de visionner n'importe quelles chaînes Twitch, la possibilité de se connecter à la plateforme pour voir toutes les chaînes que l'utilisateur suit.

**Twitch now** : Cette application permet à l'utilisateur de naviguer dans les différents streams de Twitch, c'est une application Google Chrome de Twitch, elle permet de naviguer dans Twitch sans aller sur le site. Le système de suivi est celui de Twitch, c'est-à-dire que lorsque dans l'application on suit une chaîne Twitch, dans le compte Twitch de l'utilisateur la chaîne s'ajoute également dans la liste de suivi.

**Twitch notifier** : Cette application permet à l'utilisateur d'être notifié lorsqu'un stream commence. C'est une application pour Google Chrome donc inutilisable si l'utilisateur n'utilise pas Google Chrome. C'est une application qui s'approche beaucoup de celle que je réalise car on peut saisir, dans l'application, la chaîne Twitch d'un streamer et lorsque ce streamer commence à streamer, une notification s'affiche sous forme de pop-up.

La lecture des flux vidéos se fait directement sur le site Twitch.



## Quatrième partie

# Analyse fonctionnelle

## 6 Gestion des chaînes

La gestion des chaînes Twitch reprend dans les grandes lignes les différentes fonctions de Twitch :

- Suivre / Ne plus suivre une chaîne
- Recherche de streams par nom
- Recherche de streams par popularité (nombre de spectateurs)
- Affichage du flux vidéo
- Notification par e-mail lorsqu'une diffusion commence

La gestion des chaînes de mon application a pour but de gérer les différentes chaînes. Elle se compose comme ceci :

- Liste des chaînes en ligne et hors-ligne
- Détail d'une chaîne Twitch

La liste des chaînes en ligne affiche les chaînes que l'utilisateur suit et qui sont actuellement en ligne, c'est-à-dire que la chaîne diffuse du flux vidéo en direct.

La liste des chaînes hors-ligne affiche les chaînes que l'utilisateur suit et qui sont actuellement hors-ligne. Les différentes options de configuration de chaque chaîne sont accessible depuis ces liste.

Le détail d'une chaîne Twitch consiste à afficher toutes les informations concernant une chaîne Twitch, en effet, c'est depuis là que l'utilisateur peut visualiser le stream de la chaîne, toutes les fonctions de bases de Twitch pour une chaîne sont représenté ici.

## 7 Recherche d'une chaîne

Il y a différente manière de rechercher une chaîne sur Twitch :

- Recherche par nom de chaîne
- Recherche par jeux
- Recherche par popularité

L'application reprend les mêmes filtres que le site Twitch.

## 8 Suivre une chaîne

Suivre une chaîne Twitch permet d'être notifié par e-mail lorsque la chaîne commence une diffusion en direct. La fonction "Suivre" de mon application permet de suivre une chaîne et ainsi d'être notifié lorsqu'elle commence une nouvelle diffusion en direct. Pour utiliser cette fonction, il faut obligatoirement être connecté à Twitch.

## 9 Gestion des notifications

La gestion des notifications est très simple, les notifications sont activées pour tout les streams que l'utilisateur suit. Je pars du principe que lorsqu'un utilisateur suit une chaîne, il veut être de toute manière informé d'un début de diffusion. Lorsqu'une diffusion commence, une notification apparaît sur l'écran en bas à droite avec certaines informations sur le stream.

## 10 Interface homme-machine

### 10.1 Vue principale

La vue principale permet à l'utilisateur de se connecter à Twitch et ainsi de s'informer sur les différents streams en cours. Elle permet également de rechercher une chaîne Twitch pour accéder aux détails ou au flux vidéo si elle est en train de diffuser du flux.

C'est depuis cette vue que l'on peut accéder à la configuration des notifications de chaque chaîne suivis.

Lorsque l'on réduit cette fenêtre, elle ne se réduit pas dans la barre de tâche, mais dans la barre de notification. La fermeture de la fenêtre se fait via le menu contextuel de l'icône dans la barre de tâche.

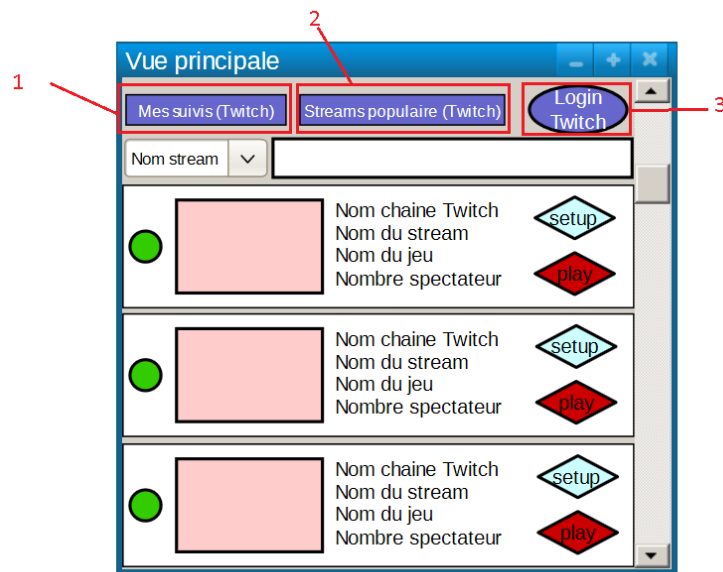


FIGURE 2 – Vue principale

1. Affiche toutes les chaîne suivis par l'utilisateur actuellement connecté
2. Affiche les streams populaire actuel de Twitch
3. Connexion à Twitch grâce à une page internet intégrée dans l'application qui s'ouvre lors du clique sur le bouton. Il faut rentrer ses identifiants pour se connecter

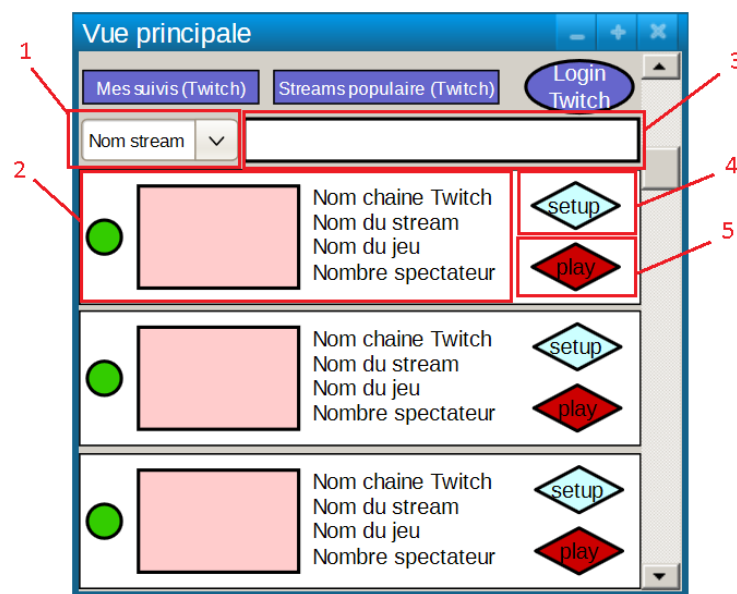


FIGURE 3 – Vue principale

1. Paramètre sur lequel la recherche va être effectuée
2. Information concernant le stream
3. Champ de saisie pour la recherche
4. Bouton pour accéder à la configuration de la chaine
5. Bouton pour lire le stream depuis l'application

## 10.2 Vue d'une notification

Cette vue apparaît lorsqu'une nouvelle diffusion commence, elle contient toutes les informations concernant la diffusion qui vient de commencer. Lorsque l'on clique sur cette vue, la vue des détails d'une chaîne s'ouvre.

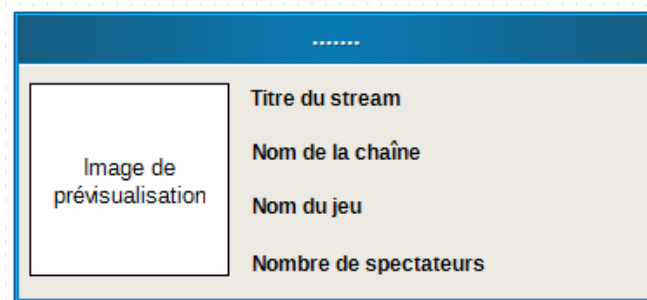


FIGURE 4 – Vue de la notification

### 10.3 Vue détail d'une chaîne

La vue des détails d'une chaîne permet à l'utilisateur connecté de regarder le flux vidéo diffusé actuellement sur la chaîne Twitch sélectionnée.

Toutes les informations concernant la diffusion actuelle ainsi que sur la chaîne en question sont affichées en bas du flux vidéo.

La réduction de la fenêtre se fait normalement.

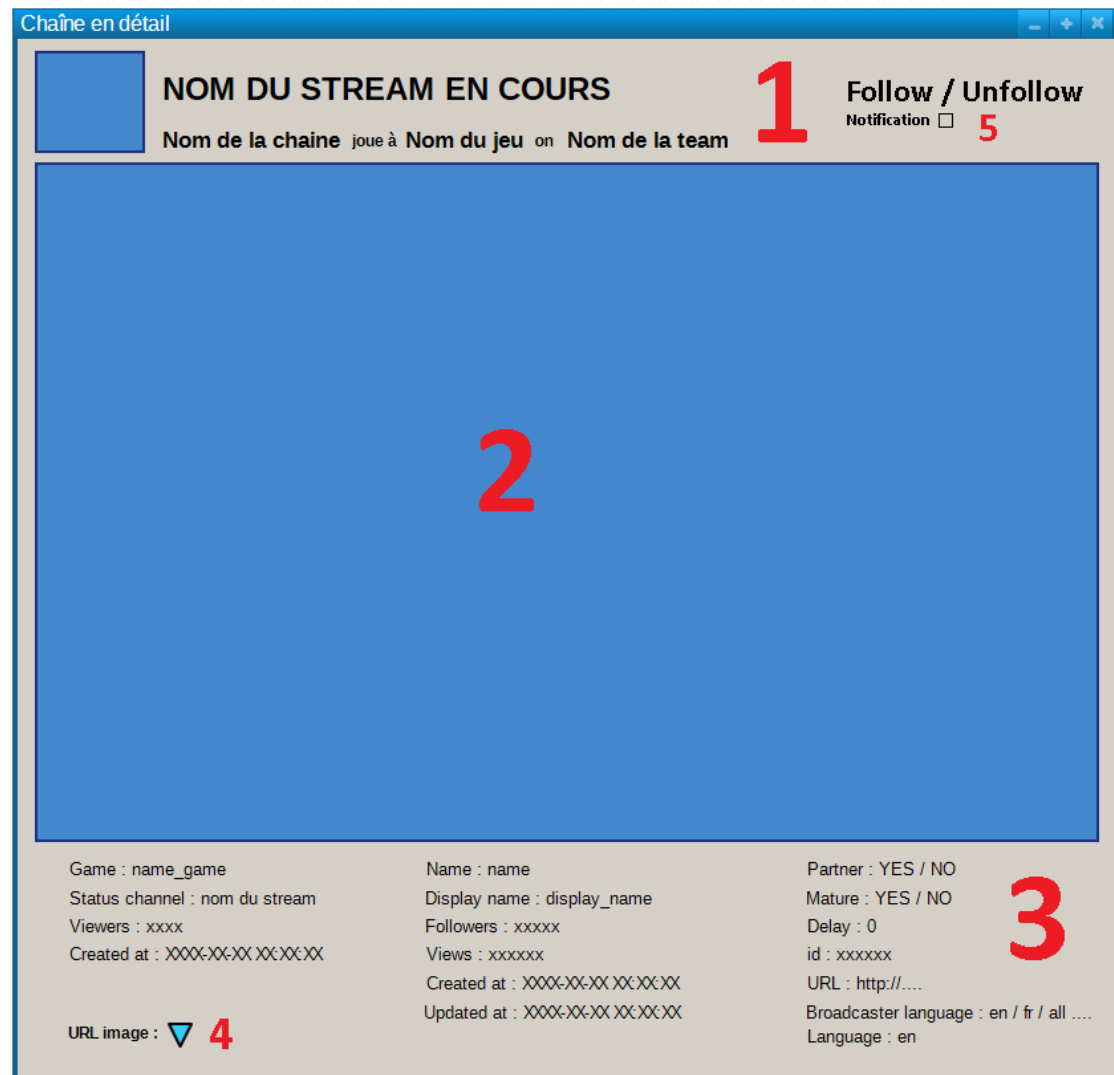
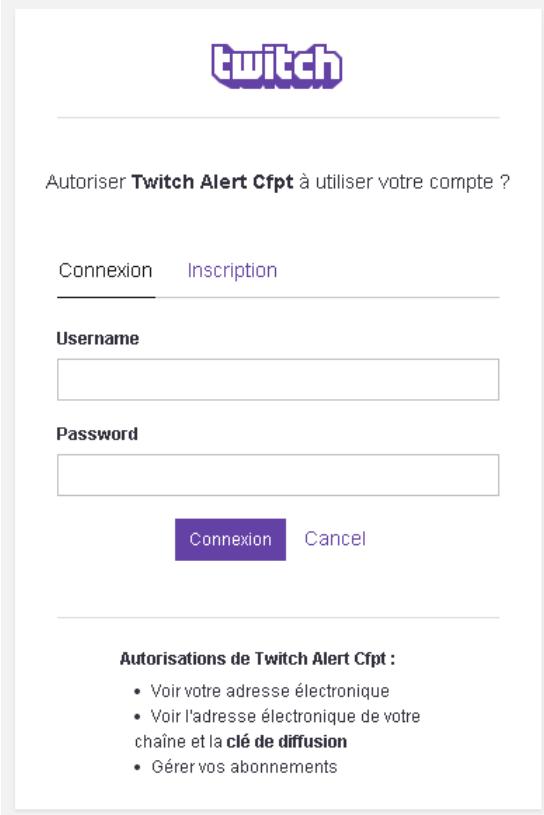


FIGURE 5 – Vue des détails d'une chaîne

1. Logo, nom de la diffusion actuelle, nom de la chaîne, nom du jeu, nom de la team(optionnel)
2. Flux vidéo de la chaîne en question
3. Toutes les informations concernant la diffusion actuelle ainsi que toutes les informations de la chaîne en question
4. Petit bouton pour afficher les images de la chaîne (logo, image de profil, image de fond ...) et leurs liens
5. Bouton pour suivre ou ne plus suivre la chaîne Twitch en question, la coche Notification s'affiche uniquement si l'utilisateur suit la chaîne et permet de recevoir ou pas les notifications de cette chaîne.

### 10.3.1 Vue de connexion

La vue de connexion permet à l'utilisateur de se connecter avec un compte Twitch à l'application.



The screenshot shows a web interface for connecting to Twitch. At the top is the Twitch logo. Below it, a question asks for authorization to use the account. There are two tabs: 'Connexion' (selected) and 'Inscription'. Under 'Connexion', there are input fields for 'Username' and 'Password'. Below these fields are two buttons: 'Connexion' (highlighted in blue) and 'Cancel'. At the bottom, there is a section titled 'Autorisations de Twitch Alert Cfpt :' followed by a bulleted list of permissions.

**twitch**

Autoriser **Twitch Alert Cfpt** à utiliser votre compte ?

Connexion **Inscription**

**Username**

**Password**

Connexion Cancel

**Autorisations de Twitch Alert Cfpt :**

- Voir votre adresse électronique
- Voir l'adresse électronique de votre chaîne et la **clé de diffusion**
- Gérer vos abonnements

FIGURE 6 – Vue de connexion à Twitch

## Cinquième partie

# Analyse organique

## 11 Diagramme de classe

### 11.1 Model

Le modèle de mon application contient trois classes : UserModel, NotificationModel et StreamModel.

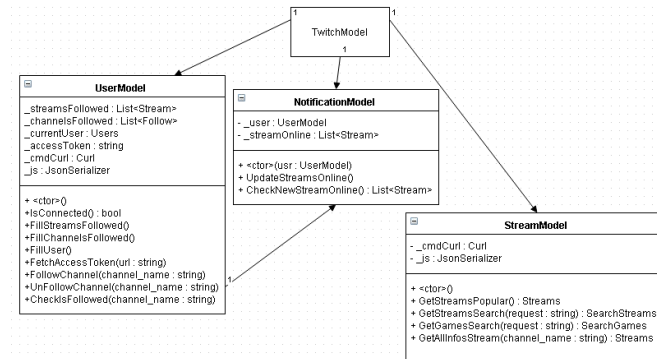


FIGURE 7 – Diagramme de classe du modèle

La classe **UserModel** contient toutes les informations et méthodes propres à un utilisateur. Quasiment toutes les fonctions de cette classe sont des requêtes HTML envoyée avec la classe "Curl" ou "JsonSerializer".

La classe **NotificationModel** contient les méthodes concernant les notifications.

La classe **StreamModel** contient les méthodes retournant toutes les informations concernant les streams en général. C'est-à-dire que les informations retournées ne dépendent pas de l'utilisateur.

## 11.2 Controllers

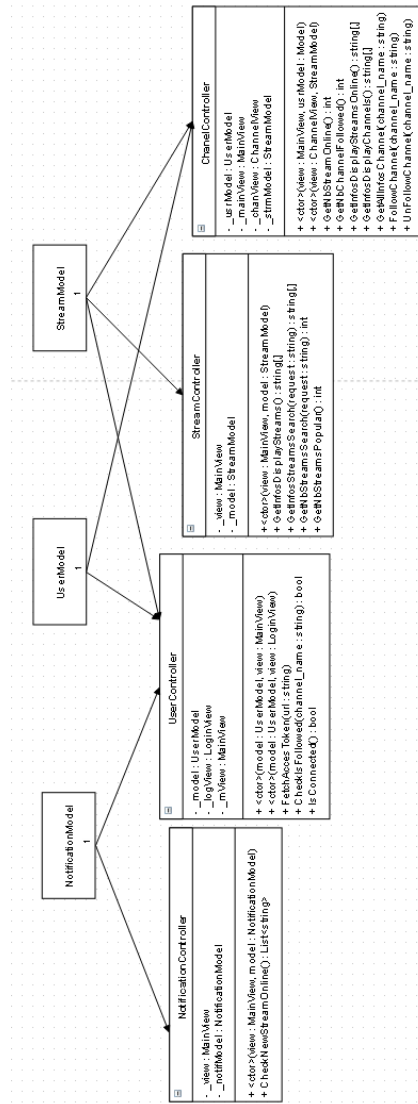


FIGURE 8 – Diagramme des contrôleurs

## 11.2.1 Views

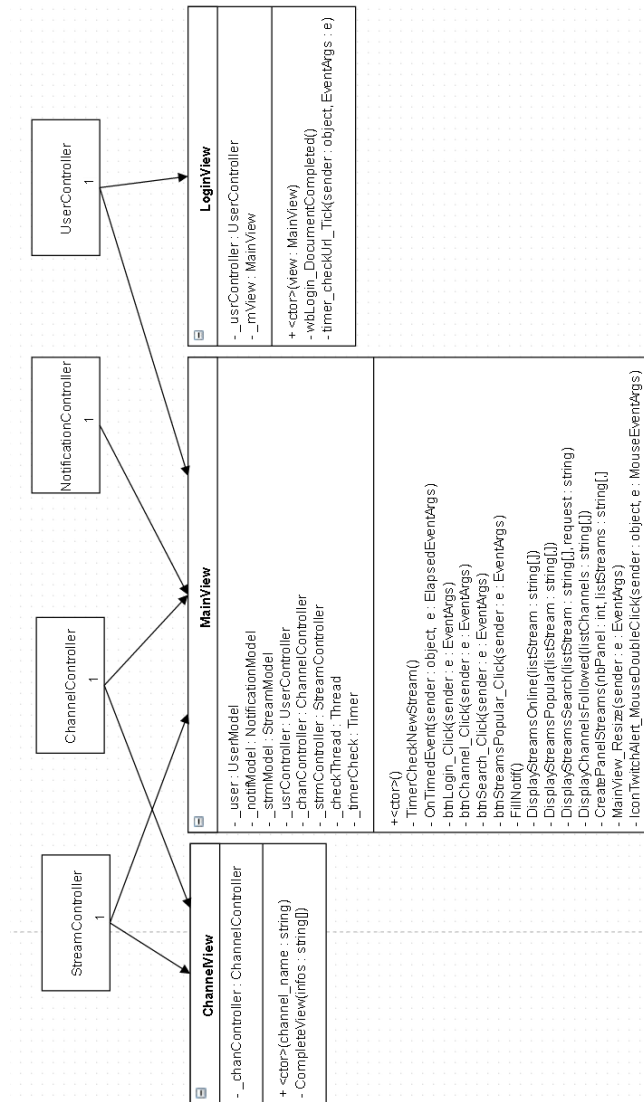


FIGURE 9 – Diagramme des vues



## 12 API Twitch

L'API Twitch sert à récupérer les différentes informations d'un flux vidéo sur Twitch et à envoyé des requêtes de modification de compte Twitch

Elle utilise un lien URL pour retrouver toutes les informations :

`https://api.twitch.tv/kraken/`

Afin de spécifier quelles informations vont être retourné par l'API, on complète la suite de l'URL. Par exemple, pour retourner les informations d'une chaîne Twitch :

`https://api.twitch.tv/kraken/channels/[nom de la chaîne Twitch]`

Toutes les informations que l'API retourne se font depuis cette page et les informations reçues sont formatées en JSON<sup>3</sup>.

L'API est composé de cette manière :

- Channels : Informations concernant une chaîne Twitch
- Chat : Informations concernant le *chat* de la chaîne Twitch
- Follows : Informations concernant les "suiveurs" de la chaîne Twitch
- Games : Informations concernant le jeu en cours sur la chaîne Twitch
- Search : Recherche des streams et des jeux
- Streams : Informations concernant le stream
- Subscriptions : Informations concernant les abonnements de la chaîne Twitch
- Teams : Informations concernant une team sur Twitch
- Users : Informations concernant un utilisateur
- Videos : Informations concernant les vidéos de la chaîne Twitchs

L'utilisation d'une application Twitch doit être validée sur le compte d'un utilisateur. En effet, avant de pouvoir utiliser un compte Twitch sur une application externe à Twitch, il faut que l'utilisateur autorise cette utilisation.

### 12.1 Récupération des données

Afin de récupérer les données reçues sur la page web de l'API, j'ai créé des classes "type" qui reçoivent les données de Twitch. Ces classes ne sont composées que de propriétés et n'ont pas de

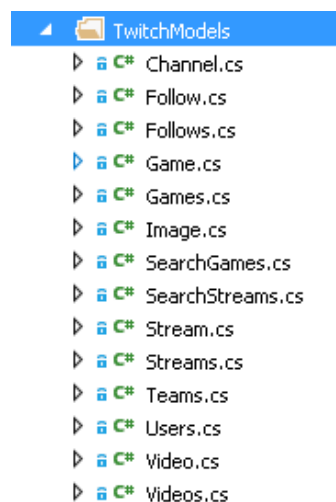


FIGURE 10 – Modèle Twitch

méthodes, chaque propriété est une donnée reçues depuis l'API. Pour spécifier que cette classe est une classe qui recevra des données sérialisées, il faut spécifier la classe en "[DataContract]" et les propriétés en "[DataMember]".

---

3. JavaScript Object Notation

Pour sérialiser les données reçues, j'ai créé une méthode générique qui retourne un objet de type générique. En effet, il faut spécifier lors de l'appel de la fonction de quel type est le résultat de la requête à l'API.

Celle-ci récupère les informations, formatée en JSON, reçues depuis la page web de l'API et les sérialise dans la classe type spécifié lors de l'appel.

```

1  /// <summary>
2  /// Serialize a Json result
3  /// </summary>
4  /// <typeparam name="T">Generic type</typeparam>
5  /// <param name="url">url for get the result</param>
6  /// <returns>variable with a generic type</returns>
7  public T Serialize<T>(string url)
8  {
9      //Create a new generic object
10     T obj = default(T);
11     try
12     {
13         WebClient wc = new WebClient();
14         //Download the result in a string
15         string strJson = wc.DownloadString(url);
16
17         //Serialize the result
18         DataContractJsonSerializer js = new ↵
19             DataContractJsonSerializer(typeof(T));
20         MemoryStream ms = new ↵
21             MemoryStream(System.Text.UTF8Encoding.Unicode.GetBytes(strJson));
22
23         //Read and add to generic object the result
24         obj = (T)js.ReadObject(ms);
25         ms.Close();
26     }
27     catch (WebException e)
28     {
29     }
30     return obj;
31 }

```

## 12.2 OAuth 2.0

Pour l'authentification d'un utilisateur Twitch sur une application externe, l'API utilise le protocole **OAuth 2.0**, 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", dans notre cas, l'application est le client.

OAuth2 définit 4 rôles :

- Détenteur des données (Nous-même)
- Serveur de ressources (Twitch)
- Le client (Twitch Alert)
- Serveur d'autorisation (Twitch)

### 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. Un token à une durée de vie limitée qui est définie par le serveur qui délivre les tokens. Un token doit rester le plus confidentiel possible, même l'utilisateur des ressources ne voit pas le token.

### Scope(portée)

Le scope est un paramètre qui sert à définir des droits sur le token. Le serveur d'autorisation propose une liste de scope disponible puis, lors de la demande d'un token, il faut spécifier les scopes à mettre en place.

### Type d'autorisation

Il existe deux types d'autorisation, celui qui nous intéresse est : L'autorisation implicite (*Implicit Grant*) L'autorisation implicite s'utilise quand l'application se trouve coté client, ce qui est notre cas. La demande se fait de la sorte :

1. L'application souhaite accéder aux données Twitch de votre compte
2. Redirigé dans un navigateur web par l'application vers le serveur d'autorisation
3. Si l'accès est autorisé, le serveur d'autorisation délivre le token et on peut le voir dans l'url.
4. On utilise le token pour faire des appels à l'API de Twitch.

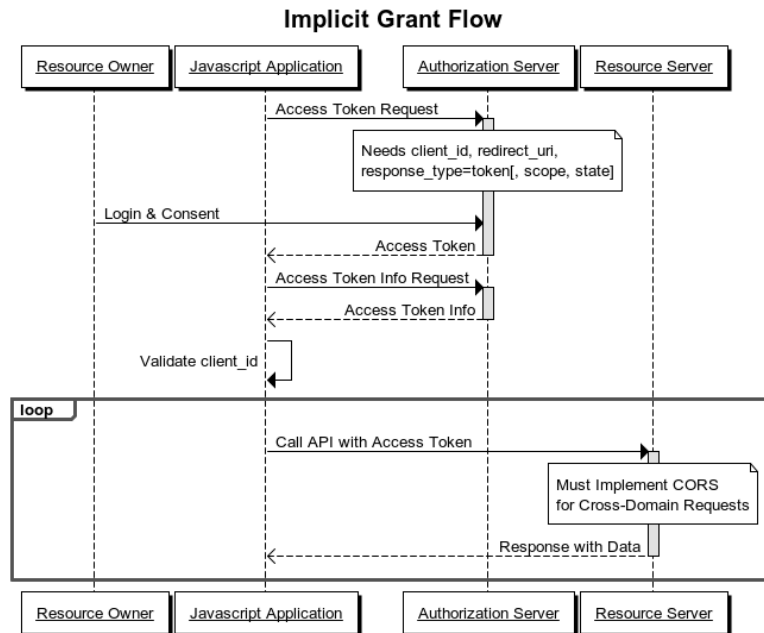


FIGURE 11 – Schéma OAuth2 : Droit implicite

### 12.3 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 partir d'une URL.

```
curl -H 'Accept: application/vnd.twitchtv.v3+json' \
-X GET https://api.twitch.tv/kraken/channels/test_user1/follows
```

FIGURE 12 – Exemple de requête cURL

Ci-dessus un exemple de requête envoyée pour récupérer les données d'une ressource.

Dans notre cas, nous envoyons une requête de type "GET" pour récupérer par exemple les informations d'une chaîne Twitch puis on reçoit les données.

L'écriture peut se faire en utilisant les commandes "PUT" ou "POST".

Les différents types de requêtes :

- GET
- POST
- PUT
- DELETE

Pour envoyer des commandes cURL à la destination souhaitée, j'ai créé une classe permettant de d'envoyer une commande et d'y recevoir la 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  /// Send a request http with cURL
3  /// </summary>
4  /// <typeparam name="T">Return generic type</typeparam>
5  /// <param name="urlRequest">request url</param>
6  /// <param name="p_method">method to use</param>
7  /// <param name="p_acces_token">access token for authenticated ↵
8  /// <returns></returns>
9  public T SendRequest<T>(string urlRequest, string p_method, string ↵
10     p_acces_token)
11 {
12     try
13     {
14         //Create a new http request
15         var httpRequest = ↵
16             (HttpWebRequest)WebRequest.Create(urlRequest);
17
18         //Init the http request
19         httpRequest.ContentType = "application/json";
20         httpRequest.Accept = "application/vnd.twitchtv.v3+json";
21         httpRequest.Method = p_method;
22         httpRequest.Headers.Add("Authorization: OAuth " + ↵
23             p_acces_token);
24         if (p_method == "PUT")
25             httpRequest.ContentType = "application/json";
26
27         //Create a http response
28         var httpResponse = (HttpWebResponse)httpRequest.GetResponse();
29
30         //Read the response
31         using (var streamReader = new ↵
32             StreamReader(httpResponse.GetResponseStream()))
33         {
34             //Add to the generics variable the result
35             T answer = ↵
36                 JsonConvert.DeserializeObject<T>(streamReader.ReadToEnd());
37             return answer;
38         }
39     }
40     catch (WebException e)
41     {
42         return default(T);
43     }
44 }
```

## 13 Site d'autorisation et de connexion

Lorsqu'un utilisateur veut se connecter et ainsi autoriser le 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/TwitchAlert/> a été mis en ligne par moi-même, mais le code a été pris de l'exemple donné par Twitch dans la documentation de OAuth 2.0.

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

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.

Pour ne pas s'attarder sur la programmation d'une connexion à Twitch, j'ai utilisé l'exemple donné dans la documentation du SDK JavaScript de Twitch. En effet, elle fournit une page web permettant l'autorisation d'une application sur un compte Twitch et la connexion à Twitch. <https://github.com/justintv/twitch-js-sdk>.

## 14 Affichage

### 14.1 Pop-up de 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>.

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 du stream ; nom de la chaîne Twitch ; nom du jeu ; nombre de spectateurs ;

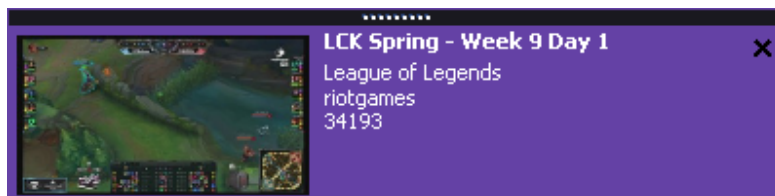


FIGURE 13 – Notification de l'application

### 14.2 Flux vidéo

L'affichage du flux vidéo à l'intérieur d'une Windows Form ce fait grâce au composant "Web-Browser". En effet, Twitch dispose d'une URL qui affiche uniquement le lecteur de flux vidéo, j'affiche donc le contenu de cette page web dans le WebBrowser, ainsi, je n'ai que le lecteur vidéo.

## 15 Notification

### 15.1 Vérification de nouveaux streams en ligne

Afin de vérifier si des streams commencent à diffuser, j'ai mis en place un timer qui va vérifier si un nouveau stream a commencé. La méthode `CheckNewStreamOnline()` de la classe `NotificationModel` sert à vérifier si une nouvelle diffusion en direct commence, elle procède ainsi :

```

1  /// <summary>
2  /// Check the new online stream
3  /// </summary>
4  /// <returns>list of new online stream</returns>
5  public List<Stream> CheckNewStreamOnline()
6  {
7  //List of new streams online
8      List<Stream> diff = null;
9
10     if(this.StreamOnline != null)
11     {
12         //Create a new list with the current online streams
13         List<Stream> oldStreamOnline = new List<Stream>(this.StreamOnline);
14
15         //Update the current online streams
16         this.UpdateStreamsOnline();
17
18         //Create a new list with the current online streams
19         diff = new List<Stream>(this.StreamOnline);
20
21         //Compare the two list
22         for (int i = 0; i < this.StreamOnline.Count(); i++)
23         {
24             for (int j = 0; j < oldStreamOnline.Count(); j++)
25             {
26                 if (this.StreamOnline[i]._id == oldStreamOnline[j]._id)
27                 {
28                     //remove if the current stream is in the old list stream
29                     diff.Remove(StreamOnline[i]);
30                 }
31             }
32         }
33     }
34     else
35     {
36         this.UpdateStreamsOnline();
37     }
38     return diff;
39 }

```

#### 15.1.1 Multi-Threading

Le coût en ressources d'un *timer* venant de la vue ralentissait la vue de l'application lors d'un *tick*. Afin de résoudre ce problème, j'ai mis en place un autre timer, celui-ci étant, par défaut lancé dans un autre processus que la vue. De ce fait, chaque fois qu'il y a un tick du timer celui ne ralentit en rien la vue. Car l'événement qui est appelé à chaque tick ne se trouve pas dans le même thread que la vue.

```

1 //Automatically excute in another thread
2 this.TimerCheck = new System.Timers.Timer();
3 this.TimerCheck.Enabled = true;
4 this.TimerCheck.Elapsed += new ElapsedEventHandler(OnTimedEvent);
5 this.TimerCheck.Interval = 5000;

```

Étant donné que l'affichage sur une vue depuis un autre Thread que celui dans lequel la vue se trouve est impossible, il faut donc encapsuler la méthode qui affiche la notification avec la méthode `Invoke((MethodInvoker)delegate)` ; qui exécute un délégué sur le thread de la vue.

## Sixième partie

## Tests

## 16 Tests fonctionnels

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
Lister les stream en ligne	Connexion à un compte Twitch Cliquer sur le bouton "Popular streams"	Affichage d'une liste de stream classé par nombre de spectateurs	Affichage d'une liste de stream classé par nombre de spectateurs
Lister les chaînes suivis	Connexion à un compte Twitch Cliquer sur le bouton "Channels followed"	Affichage d'une liste des chaînes suivis	Affichage d'une liste des chaînes suivis
Lister les chaînes suivis en ligne	Connexion à un compte Twitch Cliquer sur le bouton "Online streams"	Affichage d'une liste de stream	Affichage d'une liste de stream
Suivre une chaîne	Connexion à un compte Twitch Cliquer sur le bouton "Popular streams" Cliquer sur le bouton "Follow" de n'importe quel stream	Chaîne ajouté dans la liste des chaînes suivis	Chaîne ajouté dans la liste des chaînes suivis
Notification	Connexion à un compte Twitch Cliquer sur le bouton "Popular streams" Cliquer sur le bouton "Follow" de n'importe quel stream	Attendre quelques secondes et une notification apparait	Une notification apparait
Afficher les détails d'une chaîne	Connexion à un compte Twitch Cliquer sur le bouton "Popular streams" Cliquer sur le bouton "Play" de n'importe quel chaîne	Une nouvelle fenêtre s'ouvre avec le flux vidéo et les détails de la chaîne en question	Une nouvelle fenêtre s'ouvre avec le flux vidéo et les détails de la chaîne en question
Recherche de streams en ligne	Connexion à un compte Twitch Saisir dans l'édit text la valeur "Star" Cliquer sur le bouton "Search"	Liste les streams en ligne qui contiennent dans le nom de stream ou de chaîne "Star"	Liste les streams en ligne qui contiennent dans le nom de stream ou de chaîne "Star"
Recherche de jeux	Connexion à un compte Twitch Saisir dans le combo box "Game" Saisir dans l'édit text "Star" Cliquer sur le bouton "Search"	Liste les jeux qui contiennent dans son nom "Star"	Rien ne se passe

FIGURE 14 – Tests fonctionnels

## Septième partie

# Conclusion

### 17 Apport personnel

L'estimation de l'apport personnel apporté sur ce projet est :  
La création de la vue est 60% de moi-même, l'affichage du stream se fait via le WebBrowser, la vue de notification est un composant pris de l'externe où les propriété graphique a été modifié par moi. La gestion de notification a été faite 100% par moi.  
La méthode qui cURL a été faite 100% par moi.  
La méthode de désérialisation Json a été faite 100% par moi.

### 18 Conclusion personnelle

#### 18.1 Bilan personnel

J'ai eu plaisir de travailler sur ce projet, le fait de travailler avec une API était nouveau pour moi. De plus, travailler dans un domaine que j'affectionne tout particulièrement m'a permis de trouver une motivation sur la réalisation de ce projet. Le fait que ça soit une application que je pourrai utiliser m'a également motivé. Sur la gestion du temps, je n'ai malheureusement pas bien réparti le temps de travail tout au long du projet. En effet, la plupart du développement a été réaliser sur la fin du projet.  
Malgré tout, je suis tout de même satisfait de l'application réalisée même si elle ne remplit pas tout le cahier des charges.

### 19 Conclusion technique

#### 19.1 Bilan de réalisation

En terme de réalisation, mon application ne remplit pas le cahier des charges dans son intégralité. La partie de la gestion de notification a été négligée, en effet au fur et à mesure de l'avancement du projet, je trouvais que la gestion des notifications n'avait pas un grand intérêt. Par contre, une application fonctionnelle a tout de même été réalisé.

#### 19.2 Améliorations et perspectives

Une gestion des notifications peut être ajoutée, une gestion mieux pensée que celle proposée dans le cahier des charges. Une partie "non-connectée" de l'application qui permettrait à des utilisateurs non-connectés à Twitch d'utiliser l'application.

Pour mon travail de diplôme, je dois réaliser une application permettant la gestion des médias du Web. Il est très probable que j'utilise mon application, ou une grande partie des sources pour réaliser mon travail de diplôme.

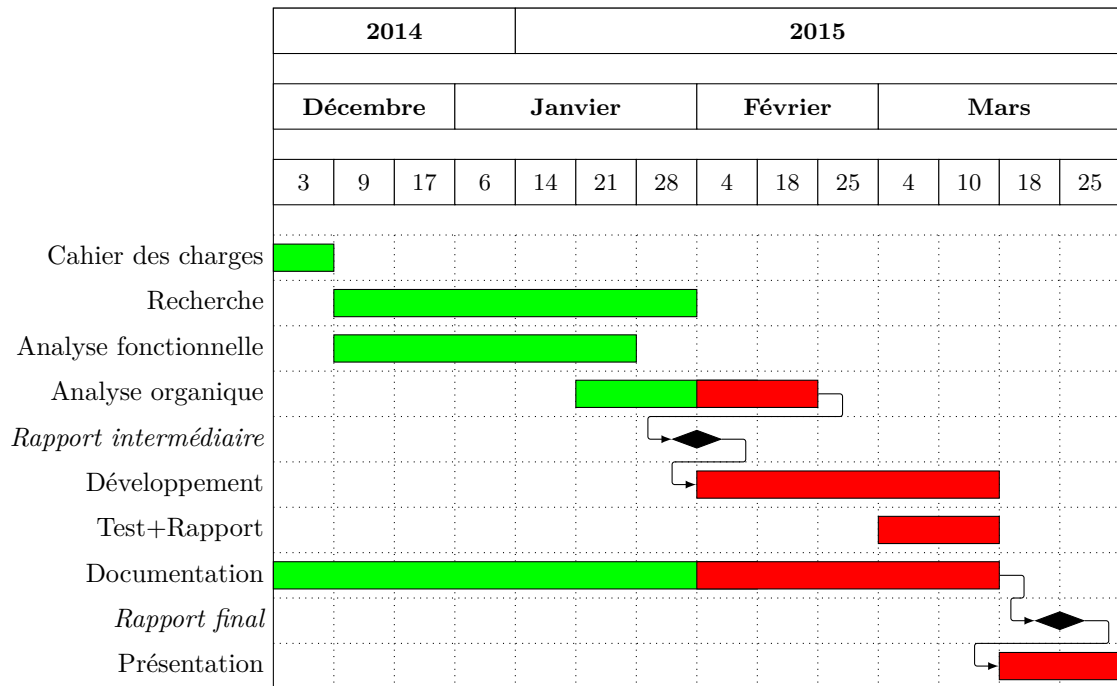


## Huitième partie

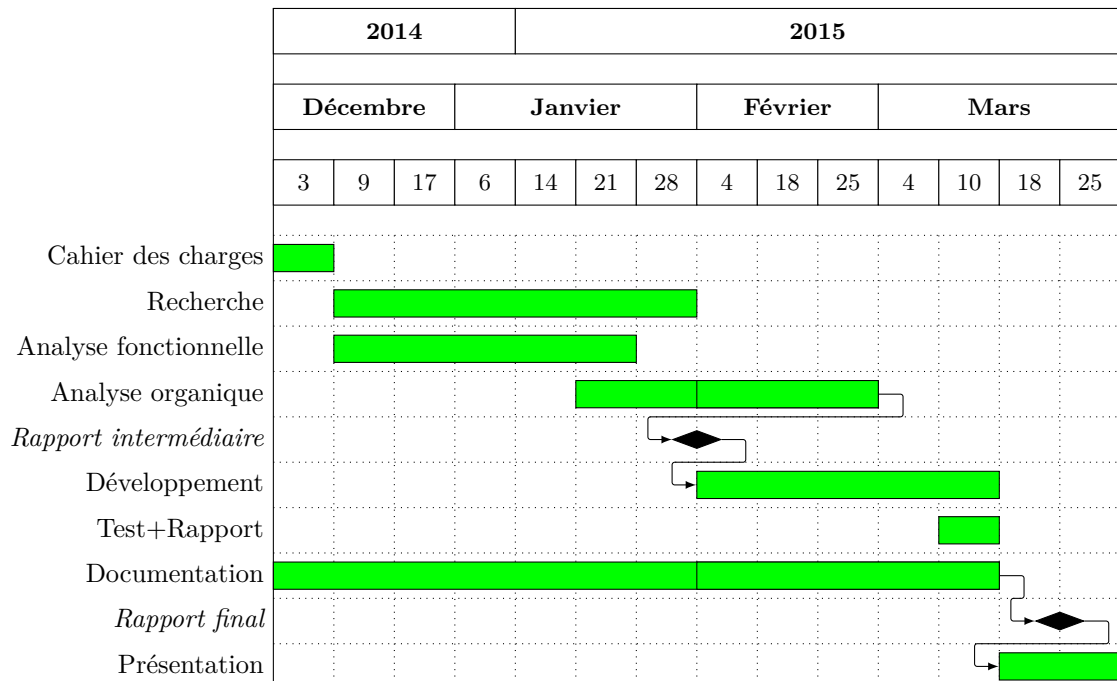
# Annexes

### 19.3 Planification

#### 19.3.1 Planification prévisionnelle



#### 19.3.2 Planning réel



## Neuvième partie

# Références

## Table des figures

1	Schema Twitch . . . . .	6
2	Vue principale . . . . .	10
3	Vue principale . . . . .	11
4	Vue de la notification . . . . .	11
5	Vue des détails d'une chaîne . . . . .	12
6	Vue de connexion à Twitch . . . . .	13
7	Diagramme de classe du modèle . . . . .	14
8	Diagramme des contrôleurs . . . . .	15
9	Diagramme des vues . . . . .	16
10	Modèle Twitch . . . . .	17
11	Schéma OAuth2 : Droit implicite . . . . .	19
12	Exemple de requête cURL . . . . .	20
13	Notification de l'application . . . . .	21
14	Tests fonctionnels . . . . .	23