# Twitch Alert

Travail de semestre

---

CFPT Informatique
Jean-Philippe Froelicher
T.IS-E2A

2014 - 2015

# Table des matières

# Première partie
# Cahier des charges

## 1 Initial

### 1.1 Titre du projet

Twitch *Alert*

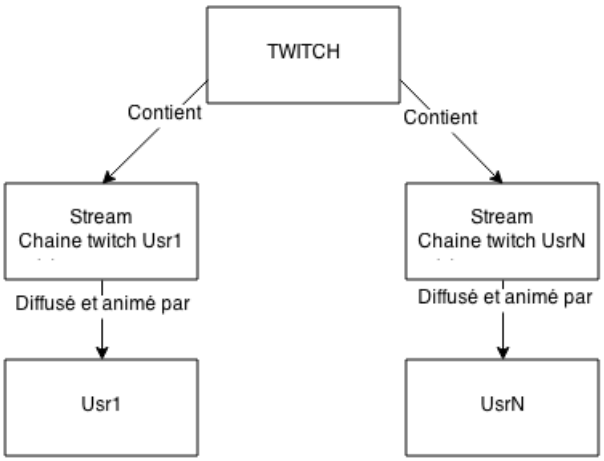
### 1.2 Objectifs du projet

Créer une application de notification(alerte) pour la plateforme de diffusion de *streaming* **TWITCH**.
Apprentissage et prise en main de l'API [1] Twitch.

### 1.3 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**

- 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

# Deuxième partie
# Étude d'opportunité

## 2 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.

### 2.1 Twitch

Twitch est une plateforme de diffusion de streaming et de VOD [2] 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 chaine Twitch de l'utilisateur en question ainsi, chaque utilisateur peut devenir si 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 leurs offrent via un services de Twitch.

#### 2.1.1 Chaîne

Une chaîne Twitch offre une multitude de possibilité pour l'utilisateur spectateur, effectivement une chaine 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 chaine. 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.

#### 2.1.2 Team

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

#### 2.1.3 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 prévenu via e-mail.

### 2.2 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 streameur 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*

### 2.3 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.

### 2.4 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.

## 3 Analyse de l'existant

### 3.1 Existant

Étant donner 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

### 3.2 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 chaines 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 chaine 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 streameur et lorsque ce streameur commence à streamer, une notification s'affiche sous forme de pop-up.
La lecture des flux vidéos se fait directement sur le site Twitch.

# Troisième partie
# Analyse fonctionnelle

## 4 Gestion des chaines

La gestion des chaines à pour but de gérer les différentes chaîne. 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 chaînes 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 chaine, toutes les fonctions de bases de Twitch pour une chaine sont reprisent ici.

## 5 Recherche d'une chaine

Il y a différente manière de rechercher une chaine sur Twitch :
- Recherche par nom de chaine
- Recherche par jeux
- Recherche par popularité

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

## 6 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.

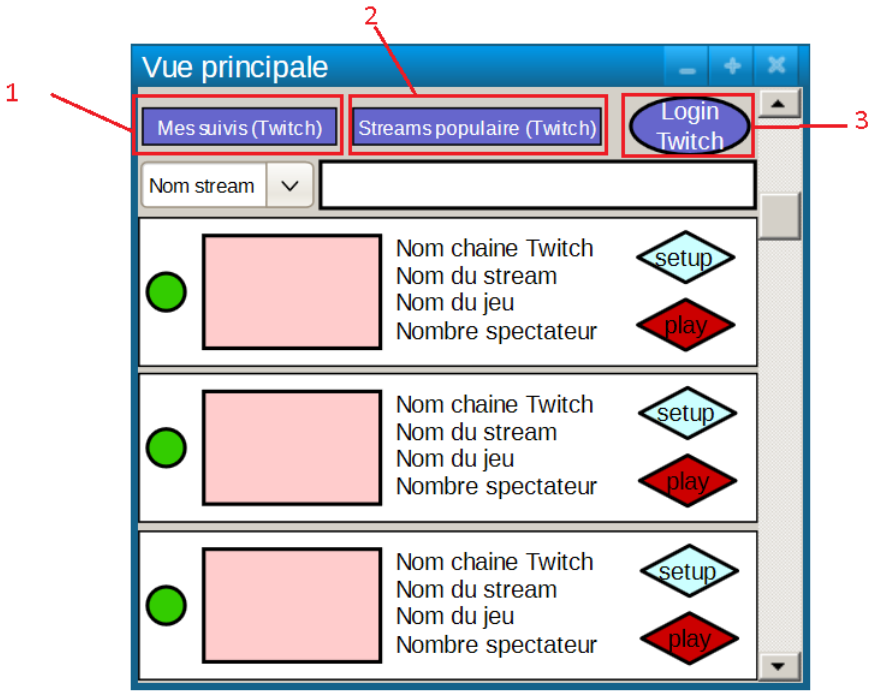
## 7 Gestion des notifications

La gestion des notifications à pour but de gérer les notifications d'un stream lorsqu'il démarre.

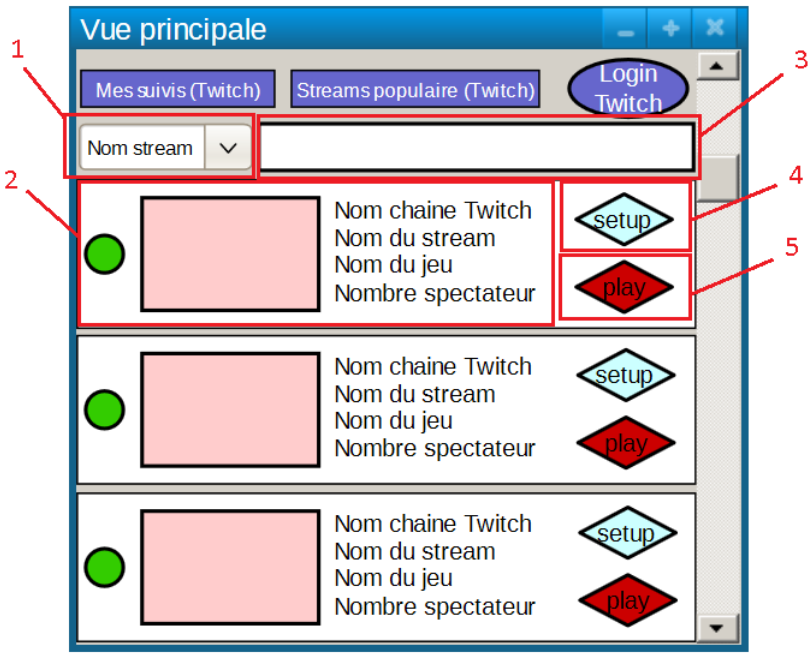
# 8 Interface homme-machine

## 8.1 Vue principale

La vue principale permet à l'utilisateur de se connecté 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 contextuelle de l'icône dans la barre de tâche.



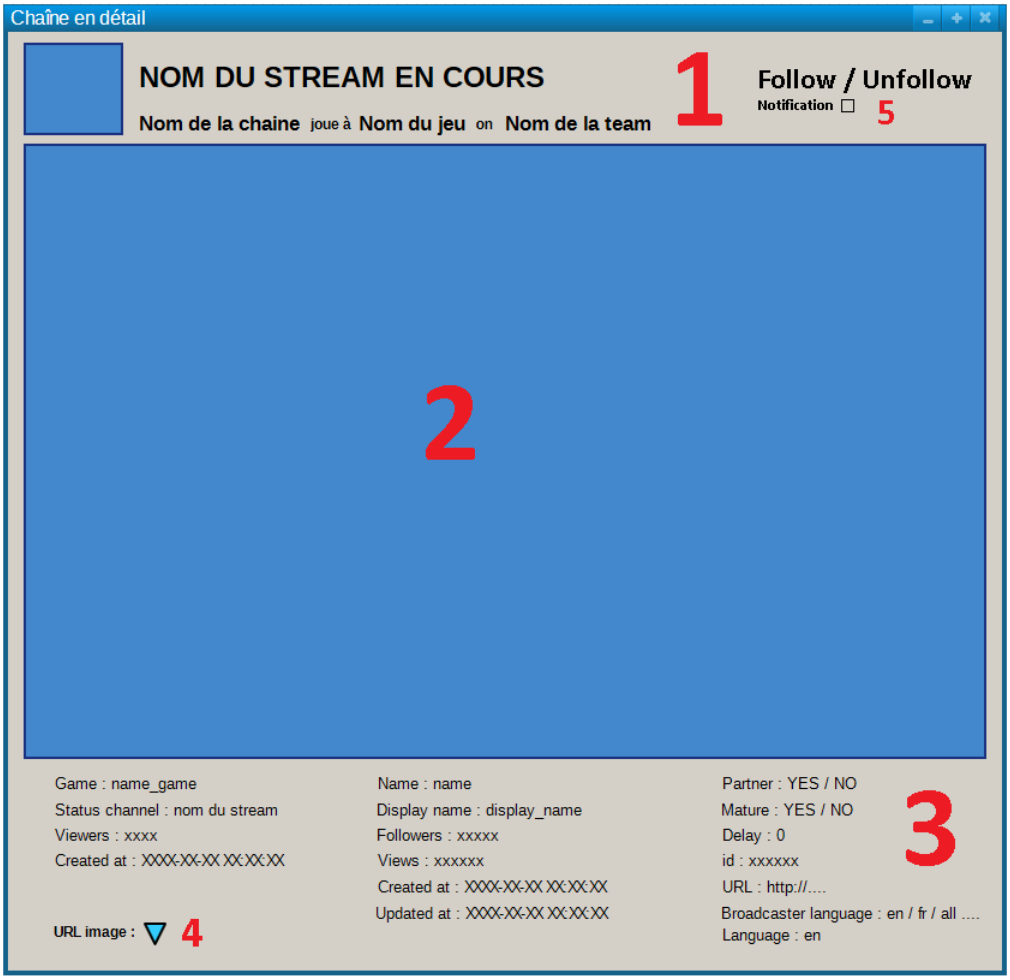

1. Affiche toutes les chaîne suivis par l'utilisateur actuellement connecté
2. Affiche les streams populaire actuel de Twitch
3. Connexion à Twitch grâce à une page internet intégrée dans l'application qui s'ouvre lors du clique sur le bouton. Il faut rentrer ses identifiants pour se connecter




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

## 8.2 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.




1. Logo, nom de la diffusion actuelle, nom de la chaine, 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.

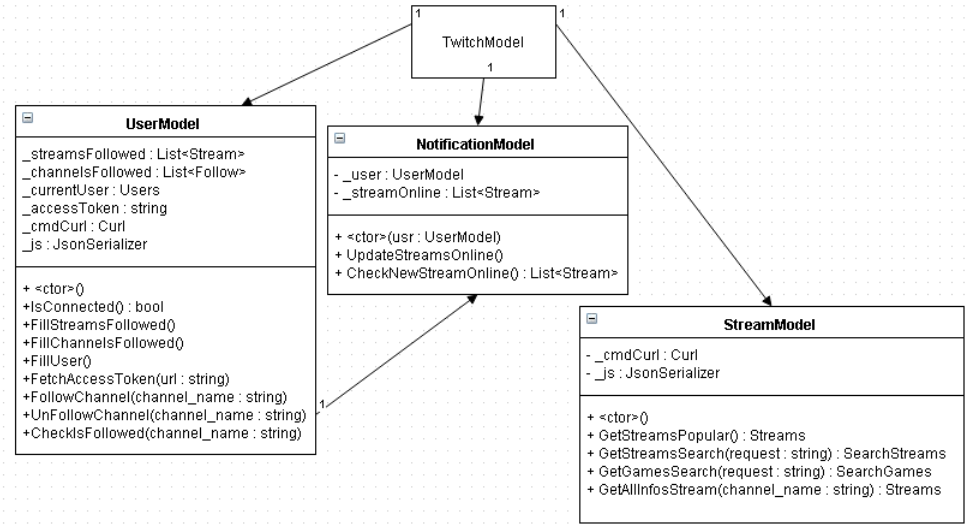
### 8.2.1 Vue de connexion

# Quatrième partie
# Analyse organique

## 9 Diagramme de classe

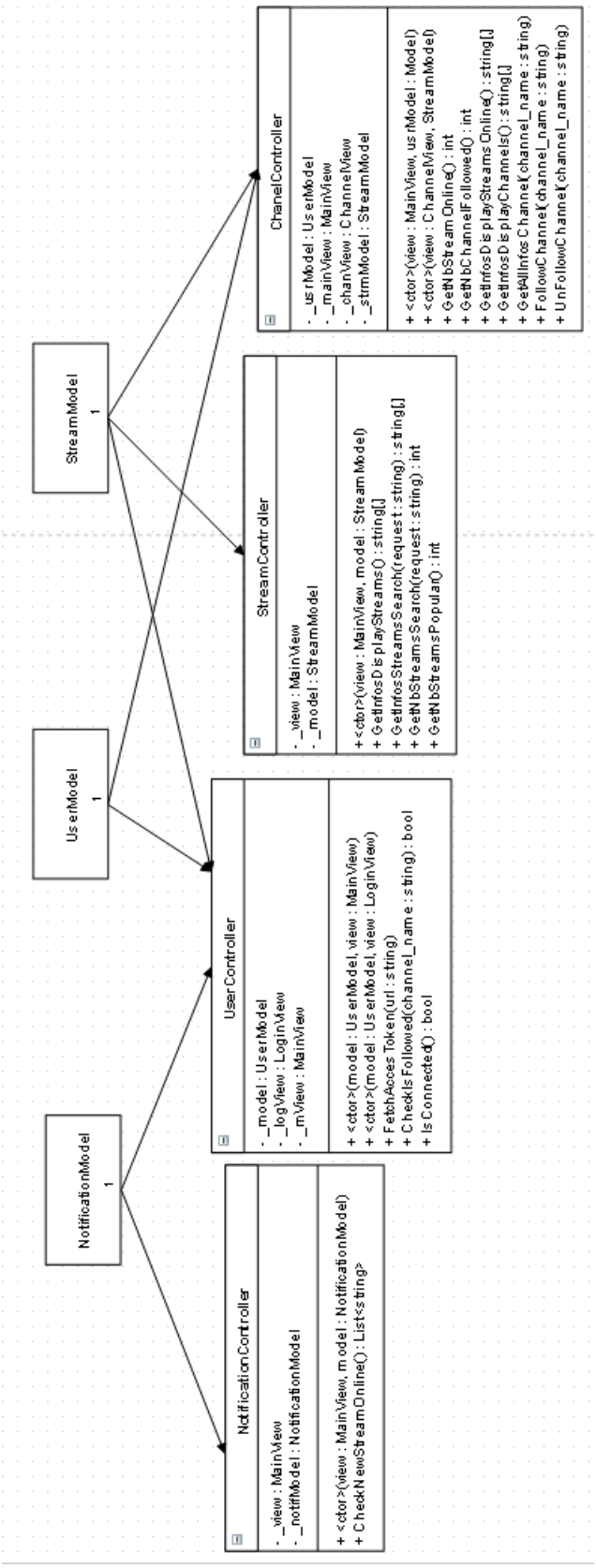
### 9.1 Model

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

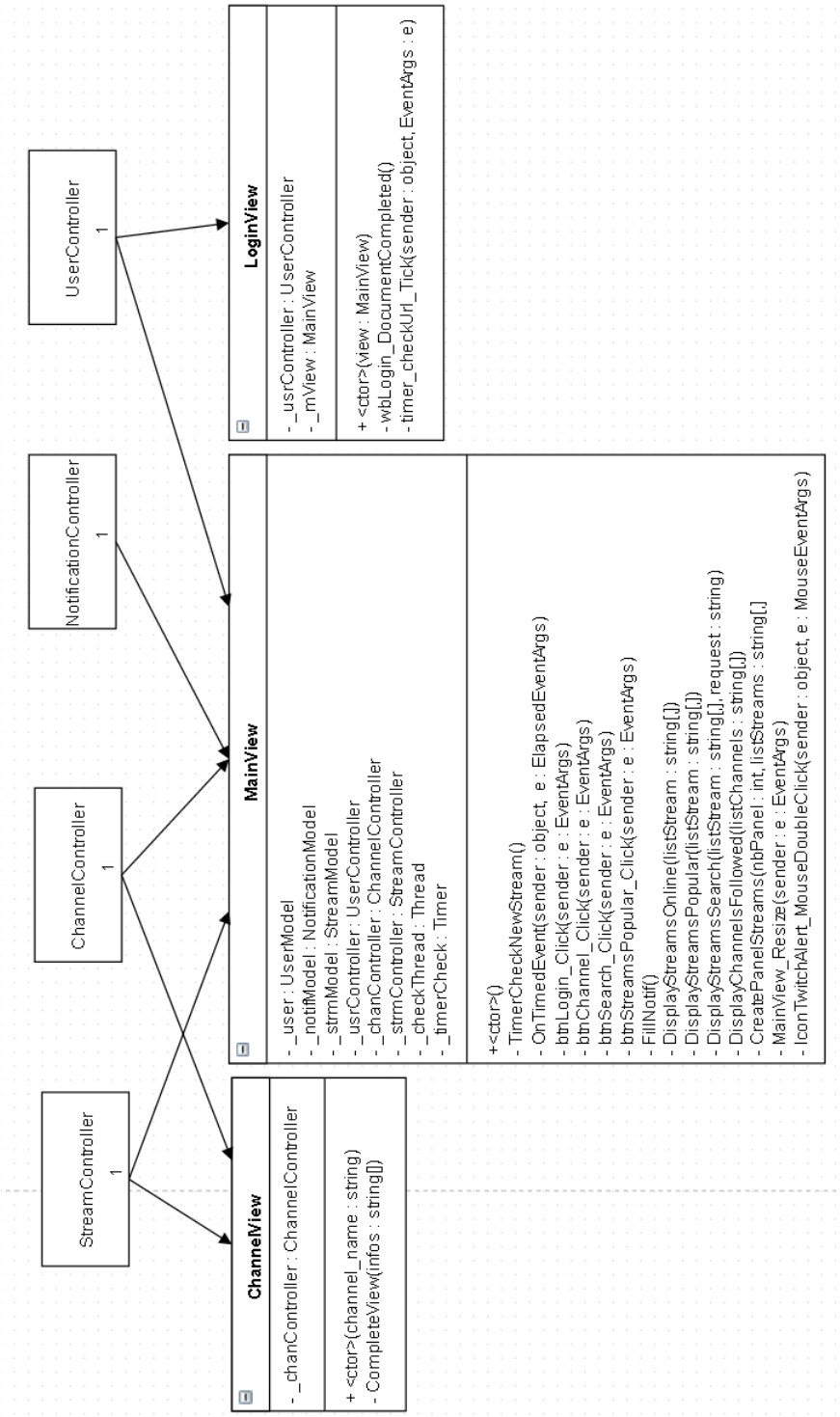



La classe UserModel contient toutes les informations et méthodes propre à un utilisateur. Quasiment toutes les fonctions de cette classes sont des requêtes HTML envoyée avec la classe "Curl" ou "JsonSerializer".
La classe NotificationModel contient les méthodes concernant les notifications.
La classe StreamModel contient les méthodes retournant toutes les informations concernant les streams en général. C'est-à-dire que les informations retournée ne dépendent pas de l'utilisateur.

## 9.2 Controllers



### 9.2.1 Views


UserController
1
NotificationController
1
ChannelController
1
StreamController
1
LoginView
- _usrController : UserController
- _mView : MainView
+ <ctor>(view : MainView)
- wbLogin_DocumentCompleted()
- timer_checkUrl_Tick(sender : object, EventArgs : e)
MainView
- _user : UserModel
- _notifModel : NotificationModel
- _strmModel : StreamModel
- _usrController : UserController
- _chanController : ChannelController
- _strmController : StreamController
- _checkThread : Thread
- _timerCheck : Timer
+<ctor>()
- TimerCheckNewStream()
- OnTimedEvent(sender : object, e : ElapsedEventArgs)
- btnLogin_Click(sender : e : EventArgs)
- btnChannel_Click(sender : e : EventArgs)
- btnSearch_Click(sender : e : EventArgs)
- btnStreamsPopular_Click(sender : e : EventArgs)
- FillNotif()
- DisplayStreamsOnline(listStream : string[,])
- DisplayStreamsPopular(listStream : string[,])
- DisplayStreamsSearch(listStream : string[,], request : string)
- DisplayChannelsFollowed(listChannels : string[,])
- CreatePanelStreams(nbPanel : int, listStreams : string[,]
- MainView_Resize(sender : e : EventArgs)
- IconTwitchAlert_MouseDoubleClick(sender : object, e : MouseEventArgs)
ChannelView
- _chanController : ChannelController
+ <ctor>(channel_name : string)
- CompleteView(infos : string[])

# 10 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 spécifié 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 fait depuis cette page et les informations reçus sont formatée en JSON [3].

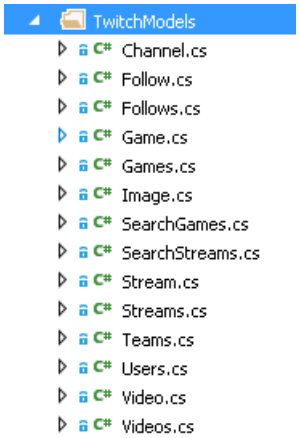
L'API est composé de cette manière :

- 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 chaine Twitchs

L'utilisation d'une application Twitch doit être validé 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.

## 10.1 Récupération des données

Afin de récupérer les données reçu sur la page web de l'API, j'ai créer des classes "type" qui reçoivent les données de Twitch.




Ces classes ne sont composée que de propriétés et n'ont pas de méthodes, chaque propriété est une donnée reçu depuis l'API. Pour spécifié que cette classe est une classe qui recevra des données sérialisé, il faut spécifié la classe en *"[DataContract]"* et les propriétés en *"[DataMember]"*.

3. JavaScript Object Notation

Pour sérialiser les données reçues, j'ai créer 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 ↩
                      DataContractJsonSerializer(typeof(T));
19                MemoryStream ms = new ↩
                      MemoryStream(System.Text.UTF8Encoding.Unicode.GetBytes(strJson));
20
21                //Read and add to generic object the result
22                obj = (T)js.ReadObject(ms);
23                ms.Close();
24            }
25            catch(WebException e)
26            {
27
28            }
29            return obj;
30        }
```

## 10.2 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.

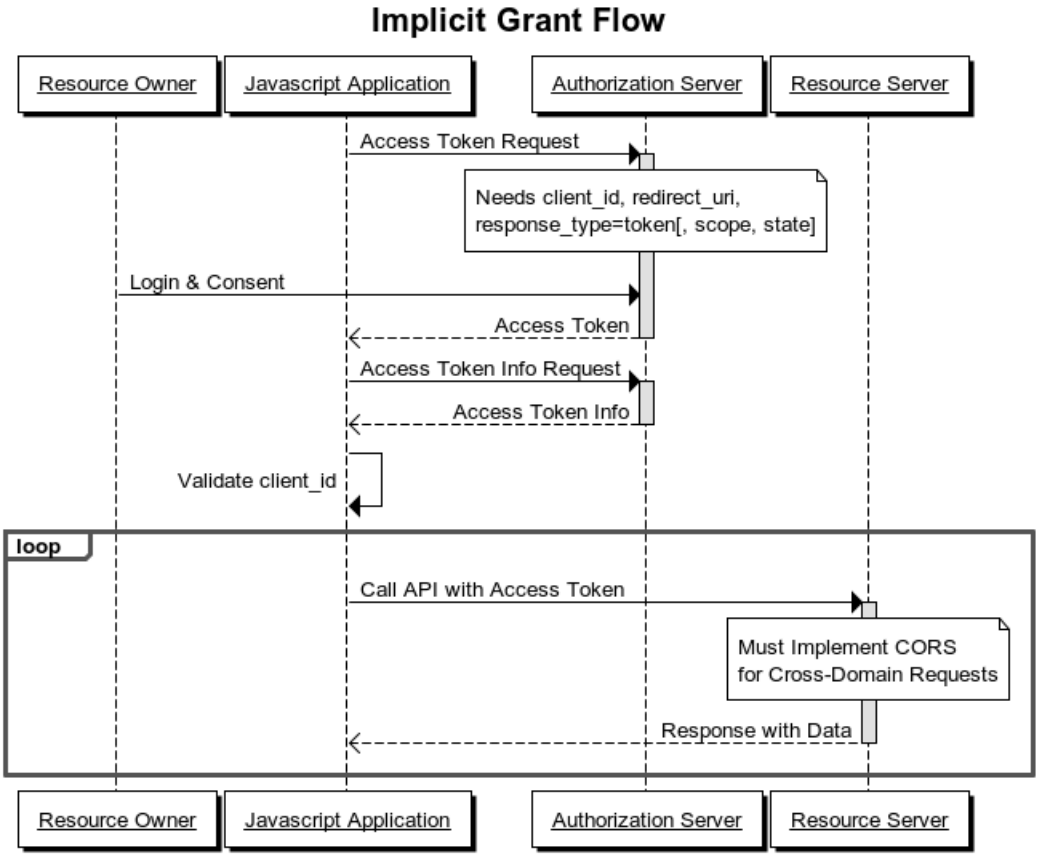
OAuth2 définit 4 rôles :
- Détenteur des données (Nous-même)
- Serveur de ressources (Twitch)
- Le client (Twitch Alert)
- Serveur d'autorisation (Twitch)

**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 limité 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 spécifié 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.



## 10.3 cURL

**C**lient **URL** **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.

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

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 reçois les données.
L'écriture peut se faire en utilisant les commandes "PUT" ou "POST".
Les différents type de requêtes :

- GET
- POST
- PUT
- DELETE

Pour envoyer des commandes cURL à la destination souhaitée j'ai créer 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 authentificated request</param>
8  /// <returns></returns>
9  public T SendRequest<T>(string urlRequest, string p_method, string p_acces_token)
10 {
11     try
12     {
13         //Create a new http request
14         var httpWebRequest = (HttpWebRequest)WebRequest.Create(urlRequest);
15
16         //Init the http request
17         httpWebRequest.ContentType = "application/json";
18         httpWebRequest.Accept = "application/vnd.twitchtv.v3+json";
19         httpWebRequest.Method = p_method;
20         httpWebRequest.Headers.Add("Authorization: OAuth " + p_acces_token);
21         if (p_method == "PUT")
22             httpWebRequest.ContentLength = 0;
23
24         //Create a http response
25         var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
26
27         //Read the response
28         using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
29         {
30             //Add to the generics variable the result
31             T answer = JsonConvert.DeserializeObject<T>(streamReader.ReadToEnd());
32             return answer;
33         }
34     }
35     catch (WebException e)
36     {
37         return default(T);
38     }
39 }
```

# 11 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 à 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.

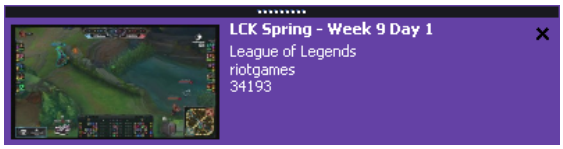
Pour ne pas s'attarder sur la programmation d'une connexion à Twitch, j'ai utilisé l'exemple donné dans la documentation du SDK JavaScript de Twitch. En effet, elle fournit une page web permettant l'autorisation d'une application sur un compte Twitch et la connexion à Twitch. https://github.com/justintv/twitch-js-sdk.

# 12 Affichage

## 12.1 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 graphique à l'interne du composant.

Les différentes informations affichées : Nom du stream ; nom de la chaîne Twitch ; nom du jeu ; nombre de spectateurs ;




## 12.2 Flux vidéo

L'affichage du flux vidéo à l'intérieur d'une Windows Form ce fait grâce au composant "WebBrowser". 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.

# 13 Notification

## 13.1 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          }
```

### 13.1.1 Multi-Threading

Le coût en ressources d'un *timer* venant de la vue ralentissait la vue de l'application lors d'um *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 ralenti 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 //Automaticaly 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.

# 14 Tests

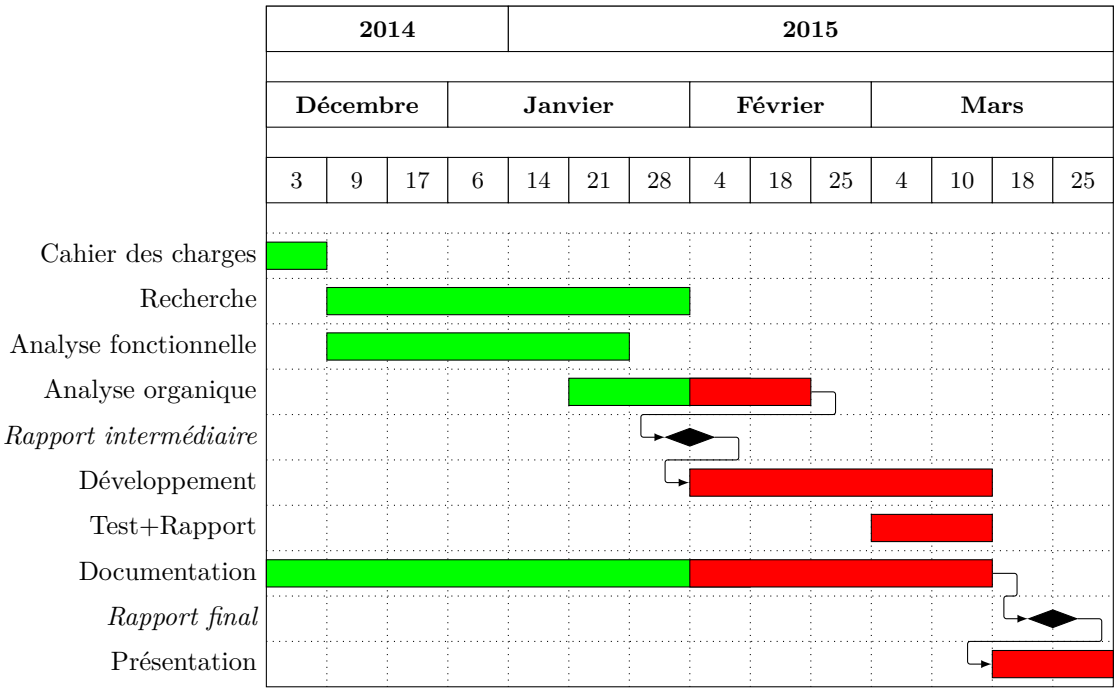
# 15 Conclusion

# 16 Glossaire

# 17 Annexes

## 17.1 Planification

### 17.1.1 Planification prévisionnelle



### 17.1.2 Planning réel