

|  |
| --- |
|  |
| **Projet Webservice Nobitsgram** |
|  |
|  |

**Auteur : Eyram DOVI**

**Professeur superviseur : Olivier liechti**

Table des matières

[1- Introduction 1](#_Toc311740911)

[2- Intérêt d’une application RESTful 1](#_Toc311740912)

[2.1-/ REST 2](#_Toc311740913)

[2.2-/ Avantages et inconvénients de REST 3](#_Toc311740914)

[2.2.1-/ Avantages 3](#_Toc311740915)

[2.2.2-/ Inconvénient 3](#_Toc311740916)

[3- Instagram 4](#_Toc311740917)

[4- Procédure d’intégration de nobits.ch aux services d’Instagram 5](#_Toc311740918)

1- Introduction

Ce travail de diplôme intervient dans le cadre de la formation de bachelor. Il a pour objectif l’intégration entre la plate-forme nobits.ch et les services fournis par instagram. La plate-forme nobits est développée à la HEIG-VD dans le cadre du projet européen "Nostalgia Bits". L'objectif de ce projet est d'explorer de nouvelles formes d'interactions sociales, avec une attention particulière aux besoins des personnes âgées. Le but est de permettre à cette population d'utilisateurs d'avoir plus d'interactions avec leur famille, mais également avec leurs voisins. Ainsi, elle permet et encourage le partage d'artifacts personnels, tels que des photos, des documents ou des objets digitalisés. Les interactions entre les utilisateurs et ce contenu (par le biais de commentaires, d'appréciations, etc.)

Dans le cadre de ce projet de diplôme, nous serons amenés à développer une application qui aura une architecture de trois tiers (couche présentation, couche métier/business, couche accès aux données) :

1.- Une application Java EE, exposant un ensemble de ressources au travers d'une API RESTful

2.- Une application PHP, gérant une couche d'accès graphique aux ressources

3.- Un module d'intégration entre le système et instagr.am, grâce au protocole OAuth.

Notre application aura pour nom Nobitsgram.

2- Intérêt d’une application RESTful

Une application RESTful est une application qui utilise les principes et concepts REST.

# 2.1-/ REST

REST **(Representational State Transfer)** est un style de programmation d’architecture distribuée orientée hypermédia. En d’autres termes, c’est un style d’architecture pour les systèmes distribués comme le World Wide Web. Ce terme REST a été proposé par Roy Thomas Fielding en 2000 lors de sa thèse (**Architectural Style and the Design of Network-based Software Architectures**) informatique présentée à l’université de Californie. Notons qu’il est à l’origine des travaux sur le protocole HTTP et sur le serveur Web Apache dont il est membre fondateur. Selon Fielding (<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> ) REST fait référence à l’image d’un système informatique web bien conçu, à partir d’un réseau de page web, permettant à l’utilisateur de naviguer dans l’application en utilisant des liens. En d’autres termes, cela veut tout simplement dire qu’un utilisateur peut lire ou modifier une ressource de l’application en utilisant une représentation de cette ressource. Ainsi, le concept essentiel de REST est la notion de ressource. Chaque information nommée d’une application peut être considérée comme une ressource : un texte brut, un document HTML/X ML, une image, une collection, etc ; bref, toute information pouvant être accédée par une référence hypertext (lien ou formulaire).

Par contre REST n’est pas un standard ni un protocole ni encore moins un format d’échange. En effet, il n’existe pas de spécification du W3C (**World Wide Web Consortium**) pour décrire l’architecture REST.

REST ne se limite pas à la réalisation d’application pour les utilisateurs que nous sommes. Il est de plus en plus utilisé dans la communication entre systèmes hétérogènes et se pose comme alternative à SOAP (même s’il est possible de combiner les deux) et au style d’architecture RPC (**Remote Procedure Call**).

REST se base sur des standards déjà existant comme :

* REST part du principe que le protocole http est largement suffisant pour l’utilisation des services web si on utilise ses méthodes GET, POST, PUT et DELETE et autre en se basant sur des ressources.
* Les URI/URL comme syntaxe universelle d’accès aux ressources.
* Les types MIME pour la représentation des ressources (text/plain, application/xml, text/xml, application/json, images ...)
* Des liens hypermédias (liens HTML) dans les documents XML, XHTML et HTML pour la représentation de la ressource et l’accès à des informations supplémentaires liées.

À partir de la définition de REST, nous comprenons qu’une application (système) RESTful est différente d’un service SOAP dans la mesure où REST repose uniquement sur l’utilisation de protocole HTTP, d’URI/URL et du langage XML, alors que SOAP utilise l’API RPC (**Remote Procedure Call**). En effet, SOAP ne suit pas les spécifications HTTP car il utilise une couche d’abstraction sur le protocole plutôt que d’utiliser de base les contraintes qu’ils imposent. SOAP est un protocole de messagerie.

Le langage informatique utilisé pour implémenter notre application est JAVA. JAVA propose une API pour l’utilisation des services REST. Cette API est JAX-RS.

# 2.2-/ Avantages et inconvénients de REST

(Source <http://fr.wikipedia.org/wiki/Representational_State_Transfer> )

## 2.2.1-/ Avantages

La thèse de Roy Fielding précise les avantages de ce style architectural par rapport à d’autres styles d’architectures d’applications web. Citons entre autres :

* L’application est plus simple à entretenir, car les liens sont mieux structurés, et de façon universelle ; les liens sont le moteur de l’état de l’application.
* L’absence de gestion d’état du client sur le serveur conduit à une consommation de mémoire inférieure, une plus grande simplicité et donc à une capacité plus grande de répondre à un grand nombre de requêtes simultanées.
* L’absence de gestion d’état du client sur le serveur permet une répartition des requêtes sur plusieurs serveurs : une session client n’est pas à maintenir sur un serveur en particulier (via une *sticky session* d’un *loadbalancer*), ou à propager sur tous les serveurs (avec des problématiques de fraîcheur de session). Cela permet aussi une meilleure évolutivité et tolérance aux pannes (un serveur peut être ajouté facilement pour augmenter la capacité de traitement, ou pour en remplacer un autre).
* L’utilisation du protocole HTTP en tirant partie de son enveloppe et ses entêtes, à l’opposé de SOAP qui ne présuppose pas un protocole : SOAP réinvente un protocole via une enveloppe, des entêtes et un document, à l’intérieur du protocole réseau l’hébergeant (la plupart du temps HTTP). il ne bénéficie donc pas des caractéristiques HTTP, gérée par l’infrastructure réseau (notamment les proxy supportant le cache côté serveur pour des performances plus importantes).
* L’utilisation d’URI comme représentant d’une ressource, permet la mise en place de serveurs cache.

## 2.2.2-/ Inconvénient

Les principaux inconvénients de REST sont la nécessité pour le client de conserver localement toutes les données nécessaires au bon déroulement d’une requête, ce qui induit une consommation en bande passante réseau plus grande. S’il est possible de coupler une application web REST à un service extérieur assurant la permanence des données lors de la durée d’une session, par exemple une base de données ou un cookie, on pourrait cependant considérer que l’utilisation d’un tel service pour gérer des données relatives à une session ouverte par le client serait en violation de la philosophie de REST. REST préfèrera l’utilisation de tableaux codés en Javascript présents dans la mémoire du navigateur client.

L’utilisation d’un formulaire HTML envoyant ses données avec une méthode comme DELETE ne peut être compris par la plupart des navigateurs. Pour pallier ce problème on émule ce comportement avec un champ caché qui transmettra le type de méthode d’envoi des données à l’application.

Par ailleurs beaucoup d’applications, bien que ne respectant pas scrupuleusement toutes les contraintes de l’architecture REST, sont largement inspirées par elle.

3- Instagram

Instagram est un réseau social grand public de masse alimenté uniquement avec des outils mobiles (tablette iPad, smartphone iPhone) via une appli iOS gratuite. Média social de nouvelle génération s'affranchissant d'un Web de bureau (ordinateur fixe ou portable). Il permet de publier, "liker" ("aimer"), commenter et partager des images (effets de filtres ou non) avec des connaissances (amis) en mode public ou privé. De plus en plus utilisé par les médias et les personnalités, Instagram séduit également le monde de l'entreprise et des institutions en termes de communication et marketing. Ce réseau social est aussi marqué par des codes et des pratiques des utilisateurs : instameets (rencontre d'usagers d'Instagram), utilisation des mots-clés par centres d'intérêts, concours, API permettant la création de produits dérivés ou d'opportunités d'affaires...

Les développeurs d’Instagram ont mis à disposition une API à destination de tous les développeurs tiers intéressés par leur service. L’intégration de la plate-forme nobits.ch aux services fournis par Instagram se fera au travers de cette API. La documentation de l’API d’Instagram se trouve sur à l’adresse <http://instagram.com/developer/> .

4- Procédure d’intégration de nobits.ch aux services d’Instagram

Avant de pouvoir intégrer toute application aux services d’Instagram, la première des choses à faire est d’enregistrer la dite application sur le site d’Instagram à l’adresse <https://instagr.am/accounts/create/?next=/developer/register/> afin de recevoir le client\_ID et le client\_secret de l’OAuth. En effet, Instagram utilise le protocole OAuth qui permet l'authentification à une API sécurisée d'une façon simple et standard depuis son bureau ou une application web tout en protégeant les noms d’utilisateurs et leur mot de passe.

# 4-1 Procédure d’enregistrement de l’application sur Instagram

La procédure pour recevoir le client\_ID et le client\_secret de l’application est décrite ci-dessous.

Le premier enregistrement concerne la création d’un compte client Instagram.

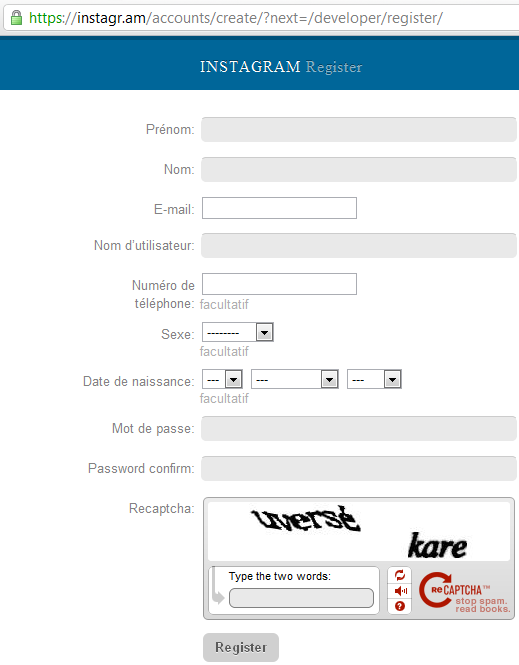


Figure 1 Création d'un compte client Instagram

Après avoir validé cette première page d’enregistrement, nous accédons à un deuxième formulaire qui se présente comme ci :

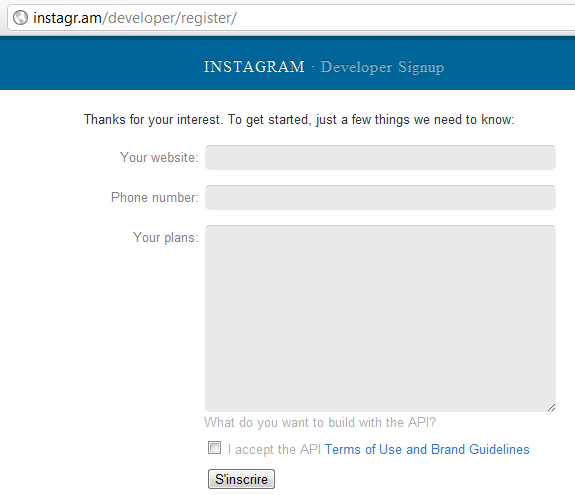


Figure 2 Formulaire d'enregistrement du développeur de l’application

Après avoir validé ce formulaire, l’utilisateur (développeur) accède à sa page client qui a cette apparence ci en sélectionnant le lien « Manage » :

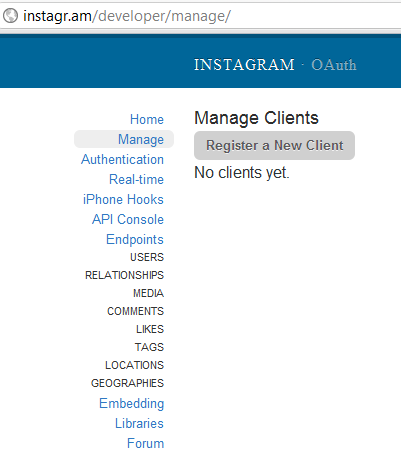


Figure 3 Page d'accueil client après enregistrement

Pour finalement recevoir son client\_ID et son client\_secret, l’utilisateur (développeur) doit cliquer sur le bouton « Register a New Client » pour accéder au dernier des trois (3) formulaires qui se présente comme ci :

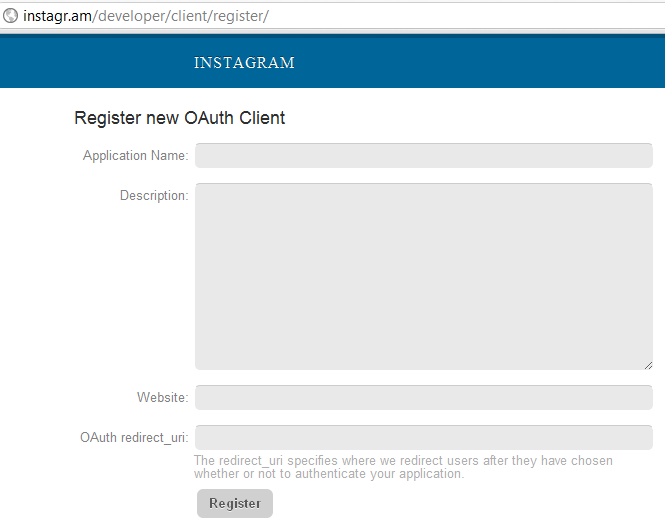


Figure 4 Formulaire d'enregistrement d'une application pour l'utilisation de l'OAuth d'Instagram

Après cette dernière étape, nous recevons finalement notre client\_ID et notre client\_secret. Les données clients se trouvent sur le lien « Manage » et se présente comme ci :

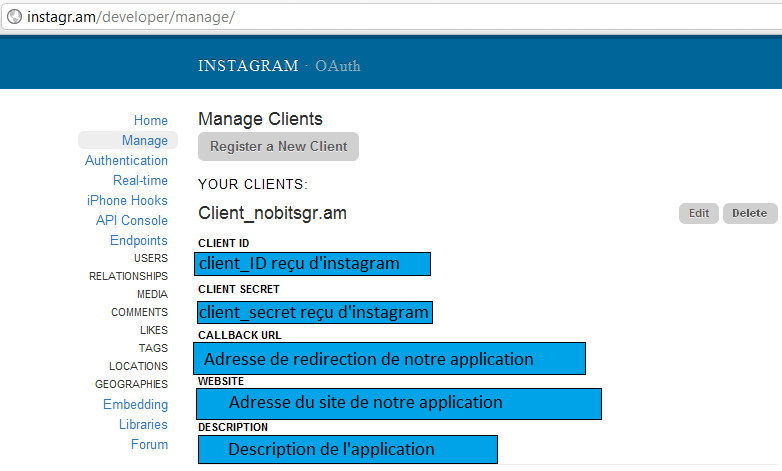


Figure 5 Affichage des données à utiliser pour l'authentification OAuth d'instagram

# 4.2- Procédure d’authentification sur Instagram

L’étape suivante après la réception du client\_ID et du client\_secret, est l’authentification au travers du protocole OAuth de la version 2.0. Cette authentification voudrait tout simplement dire que tout client qui utilisera notre application ne confie pas son mot de passe Instagram à notre application mais qu’il autorise notre application à se connecter sur compte Instagram du client. Ainsi, notre application demande à l’utilisateur de s’authentifier chez Instagram et d’autoriser le site à utiliser ses données.

La procédure d’authentification sert à récupérer l’ « acces token » Instagram de l’utilisateur. En effet, tout utilisateur possédant un compte instagram, possède un « access token » unique. Cet acces token défini chaque utilisateur. Il permet d’accéder aux données du client et si possible de modifier certaines de ses données. Notons que chaque client aura uniquement accès à son access token et qu’il ne sera pas partagé avec une tierce personne.

La récupération de l’access token est l’une des parties importante du projet.

## 4.2.1- Récupération de l’access token

La récupération de l’access token nécessite deux (2) choses.

Une première chose est de rediriger l’utilisateur de notre application (celui qui ne s’est pas encore enregistrer sur nobitsgram) vers l’url d’autorisation d’instagram à l’adresse suivante : <https://api.instagram.com/oauth/authorize/?client_id=CLIENT-ID&redirect_uri=REDIRECT-URI&response_type=code>

* CLIENT-ID = client-id reçu par Instagram
* REDIRECT-URI = adresse de redirection de notre application
* Response\_type = type de réponse que l’on souhaite recevoir d’Instagram. Nous avons 2 types de réponse, « code » et « access\_token ».

La deuxième chose est la redirection du client selon le type de réponse choisie. Une réponse de type = « access\_token » est qualifiée de « Implicit flow » et de type = « code » qualifié de « Explicit flow ».

* **Implicite flow**

En choisissant ce type de réponse, Instagram ajoute directement l’access\_token à la fin de notre adresse de redirection (dans notre cas à la fin de l’url de nobitsgram) et précédé du caractère « # ». Ce type de réponse est utilisé pour les applications ne possédant pas de serveur interne. Le choix de ce type de réponse ne permet pas la récupération directe de l’access token par le serveur. Dans ce cas précis, Instagram communique directement avec le « browser » du client en lui envoyant directement l’acces token de l’utilisateur. Ce type de réponse ne nous convient pas car nous voulons que notre application communique directement avec l’Instagram ce qui permettrait de récupérer l’access token. Pour ce genre de réponse, Instagram propose un autre type de réponse qui est le type « code » qualifié de « Explicit flow ».

* **Explicit flow**

L’explicite flow, permet au serveur d’Instagram de communiquer directement avec le serveur de notre application. Avec ce type de réponse, Instagram envoie un code à notre application. Ce code sera utilisé pour récupérer les données du client, données dans lequel se trouve l’access token. Le code reçu doit être échangé au près d’Instagram pour récupérer les données du client. L’échange se fait au travers de la requête POST. La requête POST doit être effectuée avec les paramètres de l’application à l’adresse <https://api.instagram.com/oauth/access_token>. Voici les paramètres de la requête :

**client\_id = CLIENT-ID (celui de l’application)**

**client\_secret = CLIENT-SECRET (celui de l’application)**

**grant\_type = authorization\_code (cette valeur reste telle quelle)**

**redirect\_uri = Adresse\_de\_redirection (l’adresse de redirection de notre application)**

**code = CODE (Le code reçu d’instagram)**

Il est à noter que l’access token peut expirer. Si la requête est acceptée du côté d’Instagram, ce dernier renvoie une réponse dont le format ressemble à ceci :

|  |
| --- |
| {  "access\_token": "fb2e77d.47a0479900504cb3ab4a1f626d174d2d",  "user": {  "id": "1574083",  "username": "snoopdogg",  "full\_name": "Snoop Dogg",  "profile\_picture": "http://distillery.s3.amazonaws.com/profiles/profile\_1574083\_75sq\_1295469061.jpg"  }  } |

Après avoir reçu cette réponse, il suffit juste de faire un « parsing » pour extraire les données utiles de l’utilisateur.

Après la procédure d’authentification, l’utilisateur peut donc effectuer des actions sur l’application en utilisant les ressources stockées sur Instagram.

5- Cahier de charge de l’application

Comme nous l’avons souligné plus, le but de notre application est de réaliser une intégration d’une plate-forme nobits.ch et des services REST fournis par Instagram. En partant sur cette base, il a été défini les spécification suivante de notre application nobitsgram :

**Fonctions de base**

(US1) En tant qu'utilisateur anonyme, quand je me connecte au site, je vois la page d'accueil du site nobitsgr.am:

1.1. J'ai la possibilité de me logguer en tapant mon user id et mon mot de passe et en cliquant sur un bouton "login"

1.2. J'ai la possibilité de créer un compte en cliquant sur un lien "register"

1.3. Je vois une photo venant de instagr.am, choisie au hasard parmi les photos taggées avec le mot "nobits"

1.4. La photo est mise à jour toutes les 5 secondes

(US2) En tant qu'utilisateur anonyme, j'ai la possibilité de créer un compte:

2.1. Je dois fournir mes informations personnelles

2.2. Je peux saisir mon adresse (et grâce au service de geocoding de Google, la latitude et la longitude sont automatiquement ajoutées à mon profil)

2.3. Je peux saisir mes centres d'intérêt en saisissant des mots clés (correspondant aux tags dans instagr.am)

2.4. Je dois suivre la procédure d'autorisation OAuth supportée par instagr.am (pour que l'API puisse être utilisée avec mon compte instagr.am)

(US3) En tant qu'utilisateur, après m'être connecté avec mon user id et mon mot de passe, j'arrive sur ma page personnelle; je vois plusieurs onglets (tabs):

3.1. Je peux cliquer sur un lien pour modifier mes données personnelles (y.c. mes centres d'intérêt).

3.2. Je peux cliquer sur un lien pour me déconnecter.

3.3. Dans un onglet ("gallery"), je vois une grille de 3x3 images; sur la première ligne, je vois 3 photos choisies au hasard parmi les photos de instagr.am taggées avec un de mes centres d'intérêt; sur la deuxième ligne, je vois 3 photos choisies au hasard parmi celles de mes followers; sur la troisième ligne, je vois 3 photos choisies au hasard parmi toutes les photos que j'ai "aimé" (like)

3.4. Dans un onglet ("search"), je peux taper un mot clé dans un champs de texte et cliquer sur un bouton "find"; à chaque fois que je clique, une grille de 3x3 images et mise à jour; chaque image est choisie au hasard parmis celles retrouvées dans instagr.am en fonction du mot clé saisi; je vois aussi le nombre total de photos qui existent dans instagr.am et qui satisfont au critère de recherche.

**Fonctions sociales**

(US4) En tant qu'utilisateur, je peux cliquer sur un onglet ("friends") dans ma page personnelle. Je vois des informations sur mes contacts/amis:

4.1. Je vois la liste de tous mes contacts instagr.am qui ont un compte nobitsgr.am

4.2. Je vois la liste des utilisateurs nobitsgr.am qui ont des centres d'intérêt en commun avec moi.

4.3. Je vois la liste des utilisateurs nobitsgr.am qui sont ACTUELLEMENT connectés.

4.4. Je vois la liste des personnes qui me suivent (mes followers) et que je ne suis pas; en face de chacune de ces personnes, j'ai les 3 dernières photos postées; je peux "liker" ces photos (en cliquant sur un bouton); je peux aussi cliquer sur un bouton pour devenir un follower de la personne en question.

**Fonctions géographiques**

(US5) En tant qu'utilisateur, je peux cliquer sur un onglet ("map") dans ma page personnelle.

5.1. Je vois une carte géographique (google maps); je peux cliquer sur la carte, ce qui met à jour une grille de 3x3 images; les images affichées sont choisies au hasard et ont été prises dans un rayon de 1 km;

5.2. Je vois une autre carte géographique, où sont représentés les différents utilisateurs nobitsgr.am

Fonctions d'administration

(US6) En tant qu'administrateur du site nobitsgr.am, je peux accéder à une console d'administration (en utilisant une URL spéciale, en donnant mon user id et mon mot de passe).

6.1. C'est uniquement parce que j'ai le ROLE d'administrateur que je peux y accéder (un utilisateur normal ne pourra pas y accéder).

(US7) En tant qu'administrateur, je vois le nombre de comptes utilisateurs créés;

(US8) En tant qu'administrateur, je peux naviguer dans une liste d'utilisateurs (avec pagination, c'est à dire avec n utilisateurs par page, boutons "p")

(US9) En tant qu'administrateur, je peux "bloquer" un compte (l'utilistaeur ne pourra pas se logguer et verra un message d'erreur adéquat)

(US10) En tant qu'administrateur, je peux savoir pour chaque utilisateur:

- quand le compte a été créé

- combien de fois l'utilisateur s'est connecté depuis la création du compte

- combien de fois l'utilisateur s'est connecté pendant le mois courant

- combien de recherches ont été faites depuis la création du compte

- combien de recherches ont été faites pendant le mois courant

(US11) En tant qu'administrateur, je peux taper un mot clé et afficher la liste de tous les utilisateurs qui ont utilisé ce mot clé dans leurs centres d'intérêt

(US12) En tant qu'administrateur, je peux afficher l'historique de toutes les actions effectuées par un utilisateur donné (création du compte; connexion/déconnexion; recherches; accès aux différentes listes et cartes;)

**Fonctions développeur**

(US13) En tant que développeur, je peux utiliser une API REST exposée par nobitsgr.am:

- je peux récupérer la liste des actions réalisées par un utilisateur donné.

- je peux récupérer la liste des utilisateurs, avec à chaque fois: la data de création du compte; le nombre de connexions total; le nombre de connexions dans le mois courant; le nombre de recherches faites depuis la création du compte; le nombre de recherches faites dans le mois courant.

6- Plannification et diagramme de Gant

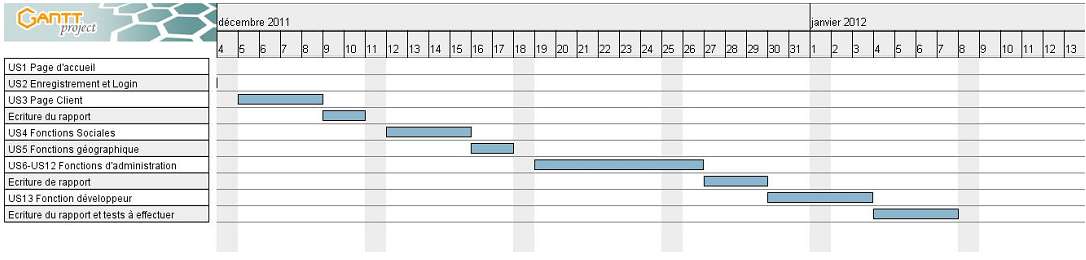


Figure 6 Diagramme de Gantt

7- Architecture de l’application nobitsgram

Notre application devra utiliser les ressources mis en service par Instagram. Nous aurons le schéma suivant pour décrire l’interaction entre nobitsgram et Instagram.