

# Projet Webservice Nobitsgram

**AUTEUR : EYRAM DOVI**

**PROFESSEUR SUPERVISEUR : OLIVIER LIECHTI**

**ANNÉE : 2011-2012**

## Table des matières

1-	Introduction .....	3
2-	Intérêt d'une application RESTful.....	4
	2.1-/ REST .....	4
	2.2-/ Avantages et inconvénients de REST .....	5
	2.2.1-/ Avantages .....	5
	2.2.2-/ Inconvénient .....	5
3-	Instagram .....	7
4-	Procédure d'intégration de nobits.ch aux services d'Instagram.....	8
	4.1-Procédure d'enregistrement de l'application sur Instagram.....	8
	4.2- Procédure d'authentification sur Instagram.....	11
	4.2.1- Récupération de l'access token.....	12
5-	Cahier de charge de l'application .....	14
	5.1- Partie « utilisateur ».....	14
	5.2- Partie « administration ».....	15
	5.3- Partie développeur.....	16
6-	Architecture de l'application nobitsgram .....	17
	6.1- Présentation des technologies utilisées.....	18
	6.1.1- Netbeans  .....	18
	6.1.2- Java  .....	19
	6.1.3- Les servlets .....	19
	6.1.4- Les JSP .....	19
	6.1.5- Serveur Glassfish  .....	20
	6.1.6- HTML .....	20
	6.1.7- CSS .....	20
	6.1.8 – JSON .....	20
	6.1.9 – JavaScript .....	20
7-	Implémentation de l'application nobitsgram .....	21
	7.1- Use Case .....	21
	7.2- Schéma de la base de données .....	23
	7.3 – Création de la base de données avec netbeans .....	24
	7.3.2 – Création du projet application web « nobitsgram » .....	24

7.3.2 – Création de la base de données .....	27
7.3.3 – création du pool de connexion.....	28
7.4 – Modèle MVC (Modèle – Vue – Contrôleur).....	29
7.4.1- Modèle .....	29
7.4.2- Vue .....	30
7.4.3- Contrôleur .....	30
8 – Interaction entre les différents modules de l'architecture MVC .....	31
8.1- Page d'accueil.....	31
8.2- Enregistrement et page d'enregistrement.....	32
8.3- Servlet gérant l'enregistrement (RegistrationServlet) .....	34
8.4 – La classe MyParser.....	36
8.5 – La classe InterrogatorInstagram.....	38
8.6 – Classe LoginServlet .....	38
8.6 – Page personnelle .....	39
8.7 – Page gallery.....	40
8.8 – La classe GalleryServlet .....	41
8.10 – classe SearchServlet .....	43
8.11- Déconnexion d'un utilisateur.....	44
8.12 – Onglet « My Account » .....	44
8.12.1 – Page settingAccount.....	44
8.12.2 – page « My Media » .....	45
8.13 – Onglet Map.....	46
8.13.1 – Page Photo Map .....	46
8.13.2 – Page Nobitsgram Map .....	48
8.14 – Onglet « Friends ».....	48
8.14.1 – Page « My contacts ».....	49
8.14.2 Page « My topics users » .....	49
8.14.3 - Page Nobitsgram users.....	50
8.14.4 – Page « My followings » .....	50
8.14.5 – Page « My Followers ».....	51
8.14.6 – Page « My Fans » .....	52
9 – Administration de Nobitsgram .....	53
10- Conclusion .....	55
11- Référence bibliographique et internet.....	56

# 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'artefacts 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.)

Le projet aura donc pour but le développement d'une application qui se chargera d'intégrer la plate-forme nobits et les services fournis par instagram. Cette application devra répondre aux exigences suivantes :

- 1- Fournir un ensemble de service et de ressource et cela au travers d'une API.
- 2- Fournir un ensemble d'interface simple et convivial à l'utilisateur afin qu'il puisse interagir avec l'application.
- 3- Fournir un module permettant l'intégration entre l'application développée durant ce projet et le instagr.am.

La seule manière de répondre à ces exigences est de développer un service web (web service). Un service web est un mécanisme de communication et d'échange de données entre des applications et des systèmes hétérogènes. Les services web fonctionnent sous le modèle de couches (n-tiers). Dans notre cas nous utiliserons les trois couches fondamentales (couche présentation, couche métier/business, couche accès aux données), d'où notre application aura pour modèle une architecture 3-tiers. Entre autre, il existe deux familles ou types de web service :

- WS- utilisant les standards SOAP(*Simple Object Access Protocol*) et WSDL(*Web Service Description Language*)
- REST (*REpresentational State Transfer*) utilisant l'architecture du web et le protocole http.

Nous utiliserons le type REST pour le développement de notre application car nous aurons à manipuler des ressources. Nous aurons finalement donc une application web service de type RESTful. En effet, toute application qui utilise le type REST est appelé application RESTful. 1.- Une application Java EE, exposant un ensemble de ressources au travers d'une API RESTful.

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 (**R**epresentational **S**tate **T**ransfer) 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 (**A**rchitectural **S**tyle and the **D**esign of **N**etwork-based **S**oftware **A**rchitectures) 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 (**W**orld **W**ide **W**ebs **C**onsortium) 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 (**R**emote **P**rocedure **C**all).

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 (**R**emote **P**rocedure **C**all). 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**

### **2.2.1-/ Avantages**

- ❖ Les applications RESTful sont de type stateless c'est-à-dire sans état. Ainsi, l'application reçoit toutes les informations nécessaires dans la requête du client et une fois la réponse envoyée, ces informations ne peuvent plus être retrouvées par l'application. Cette particularité a pour avantage la minimisation des ressources du système puisqu'il n'est pas nécessaire de garder l'état entre les appels (requêtes) du client. Ainsi, le serveur pourra traiter simultanément un plus grand nombre de requête.
- ❖ La particularité d'être de type stateless permet aussi une répartition des requêtes sur plusieurs serveurs différents. Ce procédé est utilisé par Google et Facebook pour ne citer que ceux-là. En effet, puisque l'application est sans état, le serveur traitant les requêtes ne garde pas l'état du client. Ainsi, la même requête peut être facilement traitée par un autre serveur. L'application aura donc une meilleure tolérance aux pannes. Dans le cas où un serveur se trouvant dans une localité données tombe en panne, il peut être facilement remplacé par un autre se trouvant dans une autre localité. L'évolutivité peut être aussi assuré en augmentant le nombre de serveur afin d'augmenter la capacité de traitement des requêtes (exemple de Google avec ces 900.000 serveurs).
- ❖ L'utilisation du protocole http permet aux applications RESTful de tirer le maximum des avantages de ce protocole notamment le fait qu'il soit le protocole le plus utilisé sur le web le fait qu'il soit assez simple.
- ❖ L'utilisation des liens permet un management et un entretien plus facile de l'application dans le sens où ces liens sont universel et mieux structurés ([http://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](http://fr.wikipedia.org/wiki/Representational_state_transfer)).
- ❖ L'utilisation d'URI permet plusieurs représentations de la même ressource. En effet, une ressource peut être représentée sous le format JSON, XML, TEXT, (X)HTML, CSV...
- ❖ L'utilisation d'URI permet la mise en place de serveur cache afin de filtrer les requêtes ([http://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](http://fr.wikipedia.org/wiki/Representational_state_transfer)).

### **2.2.2-/ Inconvénient**

- ❖ Les applications RESTful utilisent comme support de communication le protocole http. Il est donc clair, que les inconvénients de ce protocole se répercuteront sur les dites applications. Ainsi les problèmes liés à la sécurité du protocole http et le fait que ce protocole soit verbeux feront partie des inconvénients des applications RESTful.
- ❖ Certaines applications bien qu'elles utilisent une architecture REST, doivent néanmoins sauvegarder certaines données du client. Cela nécessite l'implémentation d'une structure pouvant sauvegarder ces données et un moyen d'y accéder. D'où l'utilisation de bases de données. L'accès aux bases de données implique une consommation plus grande de la bande passante du réseau dédié à l'application. Entre autre, il est nécessaire de garder la session de

## Projet de diplôme Nobitsgram

l'utilisateur ouverte afin qu'au cours de sa navigation entre les pages représentant les ressources de ses données, ces dernières puissent demeurer persistantes. Le seul fait de garder une session ouverte va à l'encontre même de la philosophie d'une application RESTful ([http://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](http://fr.wikipedia.org/wiki/Representational_state_transfer)).

- ❖ Autre inconvénient est le fait que le verbe DELETE ne peut être compris par un grand nombre de navigateurs ([http://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](http://fr.wikipedia.org/wiki/Representational_state_transfer)).

Même si nous notre application est de type RESTful, il est à noter qu'elle ne sera pas purement de ce type. En effet, nous aurons à sauvegarder des données clients et celles ne peuvent être fait qu'à l'aide d'une base de données et des outils de persistances pour assurer la permanence des données durant la session de chaque client.

## 3-

# Instagram



Instagram est une application gratuite sur les systèmes iOS (iPhone, iPad, iPod Touch) pour le partage de photo sur son réseau social ou sur ceux de facebook, twiter, Flickr pour ne citer que ceux là.

L'application consiste à prendre des photos et à y appliquer si on le souhaite des filtres préétablit. Le résultat est époustouflant, faisant passer une photo actuelle pour une photo datant des années 70 (effet rétro ou artistique). On peut voir l'exemple à travers cette photo prise cette année à Lausanne



Photo avec filtre "Rise"



Photo sans filtre

**Figure 1 Photo du palais de Rumine de lausanne**

L'application n'est pour le moment disponible que sur iOS et ne possède pas de site officiel sur lequel on pourrait visionner, « liker » ou commenter, bref, faire tout ce que l'on peut faire sur l'application iOS. Par contre, Instagram a bien fait les choses en mettant à disposition des développeurs des services au travers de leur API, leur permettant ainsi d'accéder à leurs données (données Instagram). Ces services ont permis l'apparition des sites comme :

- Statigram : Permettant d'avoir des statistiques (d'où son nom Statigram) concernant son compte et celui des autres.
- Followgram : site permettant de créer un bouton Instagram personnalisable et donnant un accès direct à l'URL de la galerie Instagram souhaitée (<http://www.autourdurezo.com/index.php/actus-rezo/web-20/1212-followgram-creez-un-bouton-instagram-personnalisable-pour-un-site-ou-un-blog> ).
- Webstagram, Copygram, Inkstagram, Instagre.at : site permettant de visionner les photos Instagram sur le web

L'application Instagram a reçu le prix application de l'année 2011 par Apple (Firme de l'iPhone). Instagram est de plus en plus utilisée par les médias (CNN, NBC, Washington Post), les compagnies aériennes (Air France...), les personnalités politiques (Barack Obama avec le tag #obama2012) à cause de son avantage en matière de communication et de marketing.

Vu le succès rencontré (plus de 15 million d'utilisateurs), une version est prévu pour les systèmes Android.

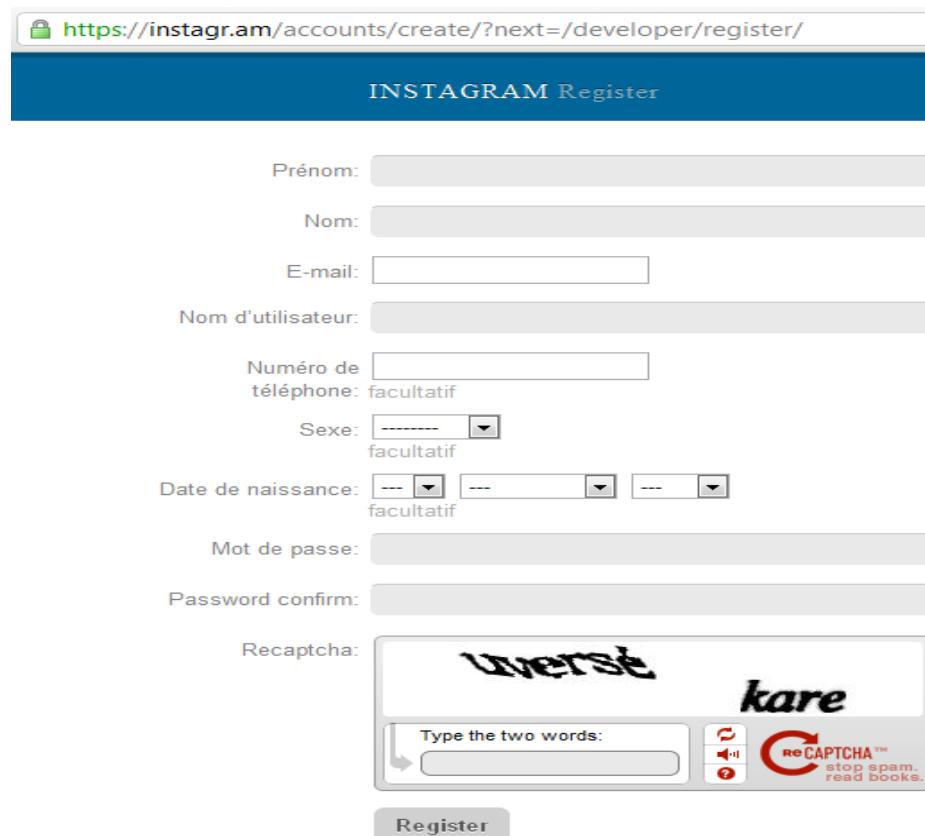
## 4- Procédure d'intégration de nobits.ch aux services d'Instagram

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.



The screenshot shows the Instagram Register page. The URL in the address bar is <https://instagr.am/accounts/create/?next=/developer/register/>. The page title is "INSTAGRAM Register". The form fields are as follows:

- Prénom: [input field]
- Nom: [input field]
- E-mail: [input field]
- Nom d'utilisateur: [input field]
- Numéro de téléphone: [input field] facultatif
- Sexe: [dropdown menu] facultatif
- Date de naissance: [dropdown menus] facultatif
- Mot de passe: [input field]
- Password confirm: [input field]
- Recaptcha: A CAPTCHA challenge with the words "unverse" and "kare" displayed above a text input field labeled "Type the two words:". It includes a "reCAPTCHA" logo and a "stop spam. read books." link.
- Register: [button]

**Figure 2** Crédit d'un compte client Instagram

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

 [instagr.am/developer/register/](https://instagr.am/developer/register/)

**INSTAGRAM · Developer Signup**

Thanks for your interest. To get started, just a few things we need to know:

Your website:

Phone number:

Your plans:

What do you want to build with the API?

I accept the API [Terms of Use](#) and [Brand Guidelines](#)

**S'inscrire**

**Figure 3 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 » :

 [instagr.am/developer/manage/](https://instagr.am/developer/manage/)

**INSTAGRAM · OAuth**

<a href="#">Home</a> <a href="#">Manage</a> <b>Manage Clients</b> <a href="#">Authentication</a> <a href="#">Real-time</a> <a href="#">iPhone Hooks</a> <a href="#">API Console</a> <a href="#">Endpoints</a> <a href="#">USERS</a> <a href="#">RELATIONSHIPS</a> <a href="#">MEDIA</a> <a href="#">COMMENTS</a> <a href="#">LIKES</a> <a href="#">TAGS</a> <a href="#">LOCATIONS</a> <a href="#">GEOGRAPHIES</a> <a href="#">Embedding</a> <a href="#">Libraries</a> <a href="#">Forum</a>	<a href="#">Register a New Client</a> <p>No clients yet.</p>
--	---

**Figure 4 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 :

<https://instagr.am/developer/client/register/>

INSTAGRAM

**Register new OAuth Client**

Application Name:

Description:

Website:

OAuth redirect\_uri:

The redirect\_uri specifies where we redirect users after they have chosen whether or not to authenticate your application.

**Register**

**Figure 5 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 :

INSTAGRAM · OAuth

Home      Manage Clients      Register a New Client

Authentication      YOUR CLIENTS:

Real-time      Client\_nobitsgr.am      Edit      Delete

iPhone Hooks      CLIENT ID: client\_ID reçu d'instagram

API Console      CLIENT SECRET: client\_secret reçu d'instagram

Endpoints      CALLBACK URL: Adresse de redirection de notre application

USERS      WEBSITE: Adresse du site de notre application

RELATIONSHIPS      DESCRIPTION: Description de l'application

MEDIA

COMMENTS

LIKES

TAGS

LOCATIONS

GEOGRAPHIES

Embedding

Libraries

Forum

**Figure 6 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. En effet, le protocole OAuth nécessite l'autorisation du propriétaire des données (ressources owner), ici, le propriétaire du compte Instagram, avant que tout échange de données ne puisse être fait entre nobitsgram (le client) et le serveur d'Instagram. Le cheminement pour l'authentification est le suivant :

- Le client (Nobitsgram) et le serveur (Instagram) doivent avoir un secret partagé (clé symétrique) pour s'authentifier. Chaque client (application web ou mobile utilisant les services de l'API d'Instagram) à une clé symétrique unique. Dans notre cas, cette clé unique est le CLIENT-ID.
- Le client (Nobitsgram) doit indiquer l'url de redirection (callback url) au serveur (Instagram).
- Le client (Nobitsgram) contacte ensuite le serveur par le biais du protocole http pour demander un token (code). Le token (code) est une suite de chiffres aléatoires qui identifiera de manière unique la requête du client. Ce code sera échangé auprès du serveur Instagram pour avoir l'access token de l'utilisateur. L'access token est une suite aléatoire de chiffre et de lettres. Chaque utilisateur (propriétaire des données ou ressources owner) de Nobitsgram a de manière unique son access token.
- Le client (Nobitsgram) va utiliser la redirection http pour envoyer le l'utilisateur (propriétaire des données) sur le site d'autorisation d'Instagram afin que le dit utilisateur donne l'autorisation au client (Nobitsgram) d'utiliser ses données (données Instagram).
- Lorsque le propriétaire donne son accord pour l'utilisation de ses données, le serveur (Instagram) signe la demande et redirige le navigateur (browser) du propriétaire vers l'url que le client à indiquer comme url de redirection.
- Le client (Nobitsgram) peut donc utiliser les données du propriétaire qui vient de donner son accord.

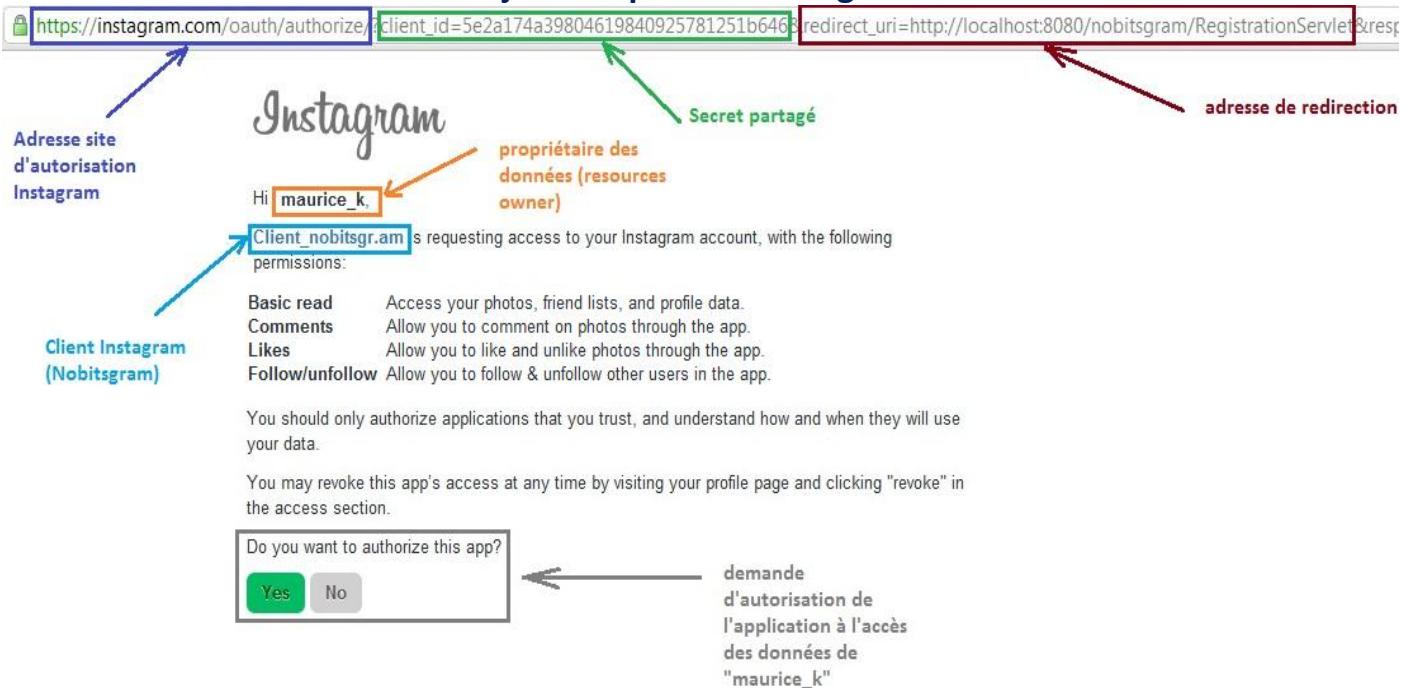


Figure 7 page d'autorisation de l'application nobitsgram

### **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](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 ».

- **Implicit 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'accès 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](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)**

La réception de l'access token se fait en ouvrant une session à l'adresse [https://api.instagram.com/oauth/access\\_token](https://api.instagram.com/oauth/access_token). Lorsque la session est ouverte, on peut envoyer les différents paramètres avec leur valeur à l'adresse ci-dessus. Lorsque l'envoie des paramètres est terminer on lui indique en faisant un flush. Tant qu'on n'indique pas au serveur instagram que l'envoie des paramètres est terminé, il attendra toujours la réception d'un paramètre. Or tant que le serveur attend un paramètre, il n'enverra jamais de réponse au client qui a effectué la requête. Il est donc nécessaire d'indiquer au serveur au travers d'un flush que l'envoie des paramètres est terminé. 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"  
    }  
}
```

Il est à noter que l'access token peut expirer.

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

---

Notre application se veut convivial et intuitive d'utilisation. Pour se faire, un certain nombre de spécifications doivent t'être mise œuvre pour définir le cahier de charge. Le cahier de charge se subdivise en 3 grandes parties. En effet, notre application est avant tout destinée aux utilisateurs d'Instagram d'où la partie « Utilisateur ». L'application doit être aussi administrée, ce qui implique la partie « Administration ». L'application ayant étant de type RESTful, est peut donc mettre à disposition des services sous formes de ressource accessible au travers de URI spécifique. Ainsi, nous avons la section « Développeur ».

### 5.1- Partie « utilisateur »

La partie utilisateur doit fournir tous les outils nécessaires et disponibles à l'utilisateur afin que l'application soit la plus conviviale et intuitive possible. Voici en bref les différentes fonctionnalités que nous aurons à développer pour les utilisateurs.

- Tout utilisateur anonyme ou non doit accéder à la page d'accueil de l'application
- Tout utilisateur à la possibilité de créer un compte tout en cliquant sur un lien ou un bouton (« Register ») prévu à cet effet.
- Sur la page d'accueil de l'application, une photo est choisie au hasard parmi celles venant de Instagram et taggée avec le mot « nobits ». Cette photo est mise à jour toutes les 5 secondes.
- Tout utilisateur possédant un compte doit pouvoir se logguer en utilisant son « username » et son mot de passe.
- Pour la création de compte, l'utilisateur doit fournir ses données personnelles et optionnellement son adresse. A partir de l'adresse fourni, les données de géolocalisation sont automatiquement ajoutées aux données personnelles de l'utilisateur.
- L'utilisateur à la possibilité de saisir plusieurs centres d'intérêts pendant sa création de compte nobitsgram. Ces différents centres d'intérêts seront eux aussi ajoutés automatiquement aux données du client.
- Pour pouvoir utiliser les données Instagram, notre application doit permettre une authentification OAuth afin que l'utilisateur puisse autoriser l'application nobitsgram à utiliser les données son compte Instagram.
- Après s'être connecté sur son compte nobitsgram, l'utilisateur doit disposer de sa page personnelle. La page personnelle présente 3 différents groupes de photos. Le premier groupe est celui des photos ayant un rapport avec l'un des différents centres d'intérêt de l'utilisateur. Le second groupe est celui des photos d'un des followers Instagram de l'utilisateur. Et le troisième groupe est celui des photos que l'utilisateur a indexé comme « like ».
- L'utilisateur à la possibilité de voir toutes ses photos et aussi de modifier ses données personnelles nobitsgram, tout cela au travers d'un lien.
- Une barre de recherche est disponible afin que l'utilisateur puisse voir une grille de 3X3 de photos différentes « taggées » avec le mot clé saisi. À chaque clic sur le bouton « find » la grille est mise à jour. Il est à noter que l'application doit afficher des informations sur le nombre de photos trouvées.

- L'utilisateur à la possibilité de voir tous ses « followers » et ses « following » se trouvant sur Instagram. Il a entre autre la possibilité de voir tous ses « fans » c'est-à-dire tous ceux qui le suivent (followers) et que lui ne suit pas (following). Entre autre, l'utilisateur peut voir tous ces contacts Instagram et aussi tous ceux qui ont un compte sur nobitsgram. L'utilisateur peut voir en temps réel tous les utilisateurs nobitsgram étant connecté. Cette fonctionnalité pourra être améliorée pour interagir (ouvrir un chat ou s'envoyer des photos) directement avec l'utilisateur connecté choisi.
- Il est aussi possible de voir selon les différents centres d'intérêt que l'utilisateur possède, ceux qui possède les mêmes que lui.
- L'utilisateur à la possibilité de « liké » une photo d'un ou de plusieurs de ses contacts (followers, followings) et même de suivre (follow) un utilisateur Instagram qu'il ne suit pas encore.
- Outre les services de bases et sociaux, l'utilisateur a à disposition deux services géographiques. Le premier consiste à l'affichage d'une map (google map de préférence) et lorsque l'utilisateur clique sur une partie de la carte, l'application affiche une grille de 3X3 photos ayant été prises sur un rayon de 1km. Le deuxième service consiste à l'affichage sur une autre map, tous les utilisateurs nobitsgram. L'utilisateur courant sera reconnaissable par une icône de couleur bleue. Les utilisateurs connectés seront reconnaissable à l'aide d'une icône de couleur verte et tous les autres à l'aide d'une icône de couleur orange.

## **5.2- Partie « administration »**

L'application nobitsgram doit pouvoir fournir à l'administrateur les moyens d'administration. Ainsi, l'application met à disposition au travers d'une URL spéciale l'accès à l'interface d'administration. Cette interface est une page web qui contiendra tous les outils d'administrations à savoir :

- le login à l'aide du mot de passe et de l'identifiant de l'administrateur,
- le blocage d'un utilisateur dans le cas où ce dernier ne respecte pas les clauses du contrat en vigueur sur nobitsgram,
- l'activation d'un utilisateur bloqué,
- l'affichage et la navigation (avec pagination) dans la liste de tous les utilisateurs
- l'affichage de toutes les données du client (date de création de compte, le nombre de connexion total jusqu'au jour courant, le nombre de connexion dans le mois, le nombre de recherches effectuées depuis la création du compte, le nombre de recherches effectuées dans le mois courant, la date de création du compte),
- l'affichage de tous les utilisateurs ayant utilisé le mot clé saisi dans une barre de recherche prévu à cet effet, dans leurs centres d'intérêt,
- l'affichage de l'historique de toutes les actions effectuées par un utilisateur données (date de création de compte, dates de connexion/déconnexion, différents termes recherchés, différents coordonnées géographiques sélectionnés, différentes listes de contact accédées).

En dernier lieu nous avons la partie offrant les services aux différents développeurs voulant utiliser l'application nobitsgram.

### **5.3- Partie développeur**

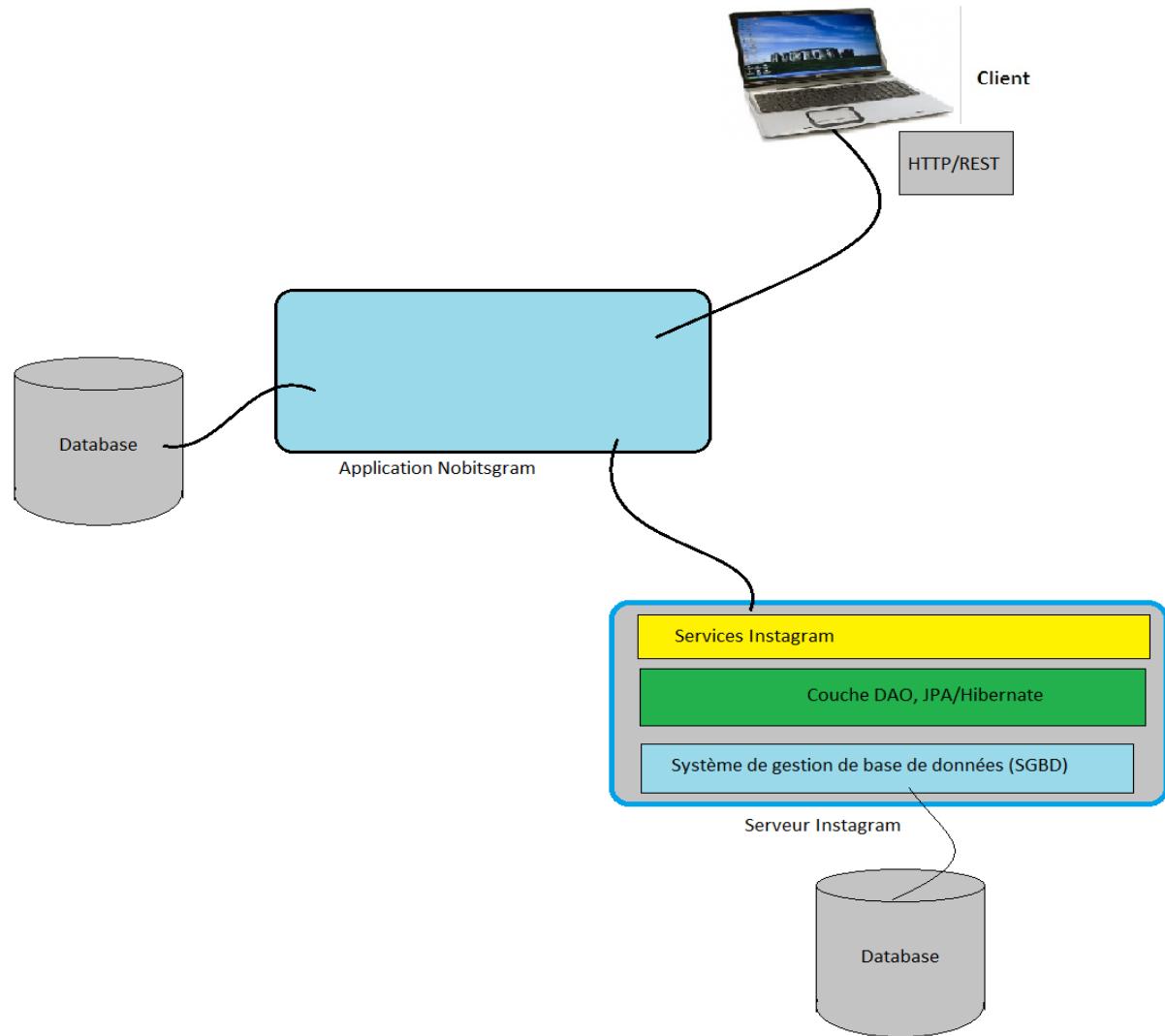
L'application se veut une application de type RESTful à l'instar d'Instagram, Google ou facebook. Elle doit donc être en mesure de fournir certains services aux différents développeurs voulant l'utiliser.

Les services fournis sont :

- La liste des différentes actions réalisées par un utilisateur donné.
- La récupération de la liste des utilisateurs ainsi que les différentes données liées à leurs activités (date de création du compte, le nombre de connexions totales effectuées, le nombre de connexions dans le mois courant, le nombre de recherches totale effectuées et le nombre de recherches effectuées dans le mois courant).

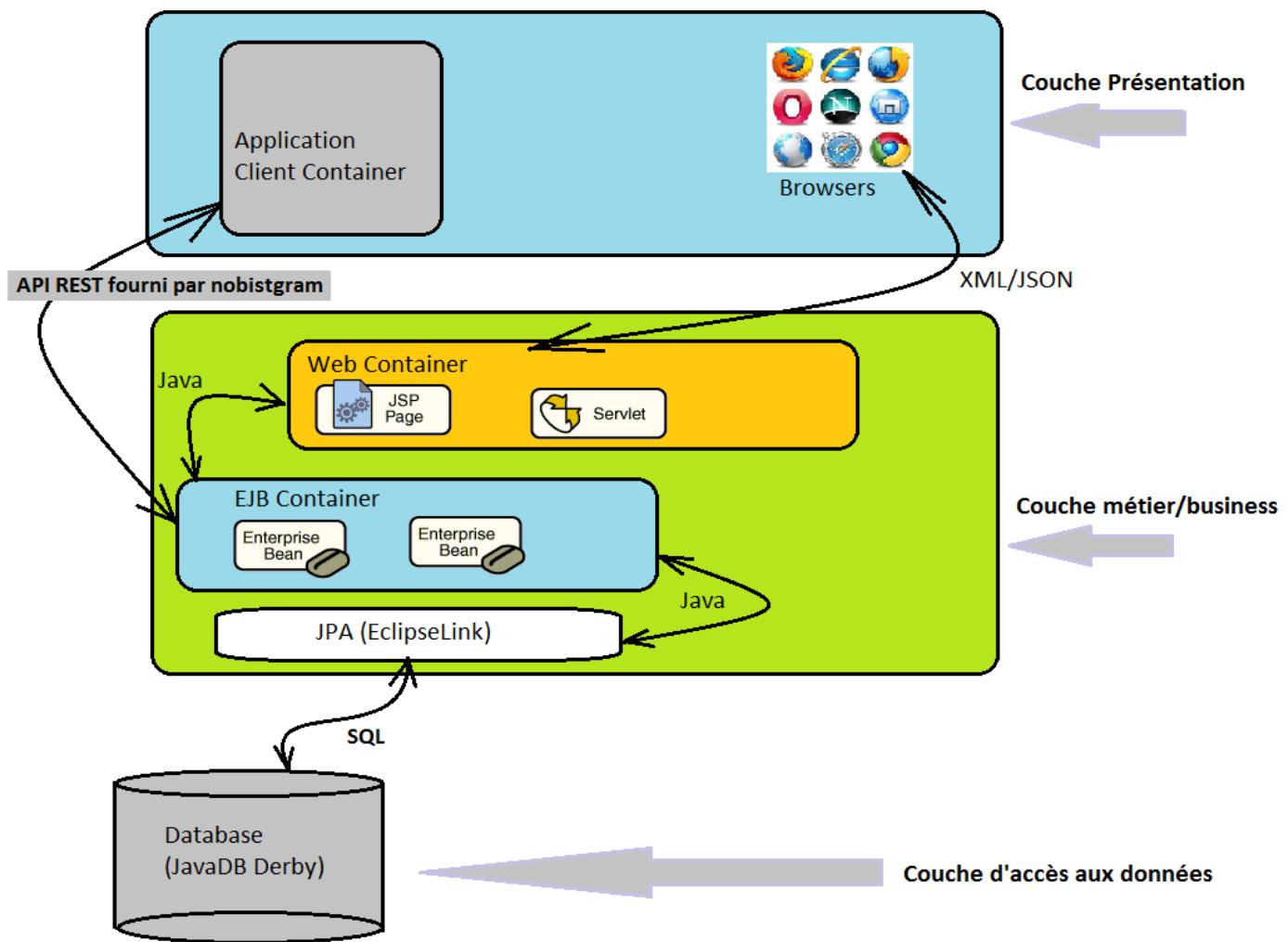
## 6- Architecture de l'application nobitsgram

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



**Figure 8 Schéma montrant l'interaction entre Nobitsgram et Instagram**

L'architecture de notre application aura l'apparence suivante :



**Figure 9 Architecture de l'application nobitsgram**

Pour mener à bien notre projet, nous utiliserons plusieurs technologies (Netbeans, java, les servlets, serveur glassfish, HTML, ...)

## 6.1- Présentation des technologies utilisées

### 6.1.1- Netbeans



Netbeans est un environnement de développement intégré (EDI ou IDE – Integrated Development Environment - ) développé et mis en open source par SUN Microsystem. Netbeans support plusieurs langage de programmation (Java, C++, C, XML, HTML, PHP, Ruby, JavaScript, Python, CSS, JSP).

Netbeans est disponible sur le différents systèmes d'exploitations notamment Windows, Linux, Mac OS. Mis à part sa gratuité, sa facilité d'utilisation, nous avons choisi Netbeans comme environnement de développement à cause de tous les avantages qu'il offre pour l'implémentation des services webs, notamment son serveur Glassfish, possibilité d'utiliser Java Persistance API, les Entreprises JavaBeans (EJB), les servlets (Java Servlet API), Hibernate, JAX-RS, etc. La version utilisée de Netbeans pour le développement de ce projet est la version stable 7.0.1. L'installation de netbeans est assez simple, il suffit d'aller sur le site à l'adresse suivante <http://netbeans.org/downloads/index.html> et de

télécharger la dernière version stable (à l'heure où l'on écrit, la dernière version est la 7.1). Après le téléchargement du fichier netbeans-xxx-ml-windows.exe (xxx représente le numéro de la version), il suffit de faire un double clic sur le dit fichier pour lancer l'installation.



### **6.1.2- Java**

Java est un langage informatique de programmation orienté objet créée par James Gosling et Patrick Naughton deux employés de Sun Microsystem, la même entreprise qui a mis Netbeans en open source. Actuellement cette entreprise a été rachetée par Oracle. L'atout majeur de Java est qu'il est portable sur plusieurs systèmes d'exploitation. Java et C++ sont assez similaires dans le sens où ce premier reprend en partie la syntaxe de C++. Avant de faire fonctionner java sur sa machine, il est nécessaire d'avoir la JVM (Java Virtual Machine) installée. L'installation de la JVM se fait au travers de l'installation du JDK (Java Development Kit). Le JDK se télécharge sur le site d'Oracle (puisque c'est l'actuel propriétaire) à l'adresse

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>. La version utilisée pour le développement de l'application nobitsgram est la version 6. Pour le cas de notre application, nous utiliserons la technologie java dédiée aux applications web. Cette technologie est Java EE (Java Enterprise Edition). Java EE contient plusieurs composantes (JSP, JPA, JTA, EJB, JNDI,...) dont celui qui nous permettra de manipuler des pages web. Ce composant est la Servlet fourni au travers de Java Servlet API. La version Java EE utilisée est Java EE6.

### **6.1.3- Les servlets**

Une servlet est une classe ou application java fonctionnant du côté serveur et permettant de créer dynamiquement des données qui seront fourni au client. Elle permet ainsi, en fonction de la requête du client de retourner une réponse qui sera sous le format (X)HTML ou xml et qui sera traduit par le navigateur client comme page web. Il est donc clair que son grand avantage est sa capacité à générer dynamiquement des pages web. La génération dynamique des pages web se fait au travers des JSP (JavaServer Page).

### **6.1.4- Les JSP**

La JSP (JavaServer Page) est une technologie Java permettant de générer dynamiquement le contenu des pages web, contenu pouvant provenir d'une base de données ou d'un traitement côté serveur. La JSP offre l'avantage qu'on peut y insérer différents type de code (Java, JavaScript, AJAX, CSS). La JSP servira de vue à notre application. Le serveur au travers duquel la JSP communiquera avec la servlet est le serveur Glassfish.



## **6.1.5- Serveur Glassfish**

Glassfish est le nom du serveur d'application de Java EE (Java Enterprise Edition). Pour le moment il est le seul serveur totalement compatible Java EE. Glassfish propose un programme serveur pour les applications web et fourni une plate-forme complète de type Apache MySQL PHP. Glassfish est administrable à partir du concept de domaines gérés par un DAS (Domain Administration Server) et des « node agent » (nœuds agents) exécutés sur une machine. Le DAS permet de gérer le système central contenant les applications déployées. L'installation par défaut (lors de l'installation de netbeans) fourni deux outils de gestion et d'administration, l'un à travers une interface graphique (<http://localhost:4848>) et l'autre à travers une interface de ligne de commande nommée **asadmin** (<http://jlafosse.developpez.com/livres/javaee/glassfish/php/#LI> ).

## **6.1.6- HTML**

HTML (HyperText Markup Language) est le type de données utilisé pour représenter les pages web. HTML utilise essentiellement des balises (<body>, <table>, <div>, <a>) et qui permet d'insérer des liens renvoyant eux-mêmes à d'autres documents HTML. On peut inclure dans HTML des ressources multimédias (vidéo, photo, son), des applets (programme java ayant une interface graphique et pouvant t'être insérée dans une page web).

## **6.1.7- CSS**

Le CSS (Cascading Style Sheets ou feuilles de Style en Cascade) est un langage informatique servant à structurer et à mettre en forme la présentation des pages web. Le CSS peut être inclus directement dans le document HTML ou XML. Mais, l'on tire tout son potentiel en le mettant dans un fichier séparé. Ainsi, plus besoin dupliquer le code, il suffit de faire référence au fichier en question et le tour est joué. L'autre avantage, est qu'il permet le maintien et la gestion des mises en forme.

## **6.1.8 – JSON**

JSON (JavaScript Object Notation) est un format de texte pour l'échange de données entre deux applications quelconques. JSON est indépendant de tout langage de programmation. La syntaxe de JSON fait qu'il est facile à utiliser et est moins verbeux comparativement au format xml. JSON est formé de trois (3) types de données structurelles, les objets (ensemble de couple nom/valeur), les valeurs (chaîne de caractère, booléen ou null, un objet, un nombre, un tableau). Les structures d'une valeur peuvent être imbriquées (<http://json.org/jsonfr.html> ).

## **6.1.9 – JavaScript**

JavaScript comme son nom l'indique est un langage de script. JavaScript est essentiellement utilisé du côté client d'une application web ou d'une page web. Il est insérer directement dans le code (X)HTML, ce qui permet de lui apporter une évolution notable. JavaScript met à disposition des écouteurs pour la souris ce qui permet de savoir quel élément de la page web a été sélectionné ou cliqué du côté du client. JavaScript permet de travailler avec des objets dans du code HTML.

## 7- Implémentation de l'application nobitsgram

Nous avons implémenté notre application sous le modèle MVC (Modèle – Vue – Contrôleur). Dans le cadre de ce projet, nous avons décidé de travailler avec l'IDE Netbeans. La version utilisée est la 7.0.1. Pour réaliser cette application, nous avons utilisé le langage Java (Servlet et classe java), du HTML (pages JSP) éventuellement du javascript (pages jsp) et du JPQL (Java Persistence Query Language) pour interroger la base de données.

Pour mieux cerner le travail à faire, nous avons réalisé un diagramme « Use Case » de l'application, ainsi qu'un schéma de la base de données. Après ces deux schémas, nous avons établi un diagramme UML des classes et de leurs interactions.

### 7.1- Use Case

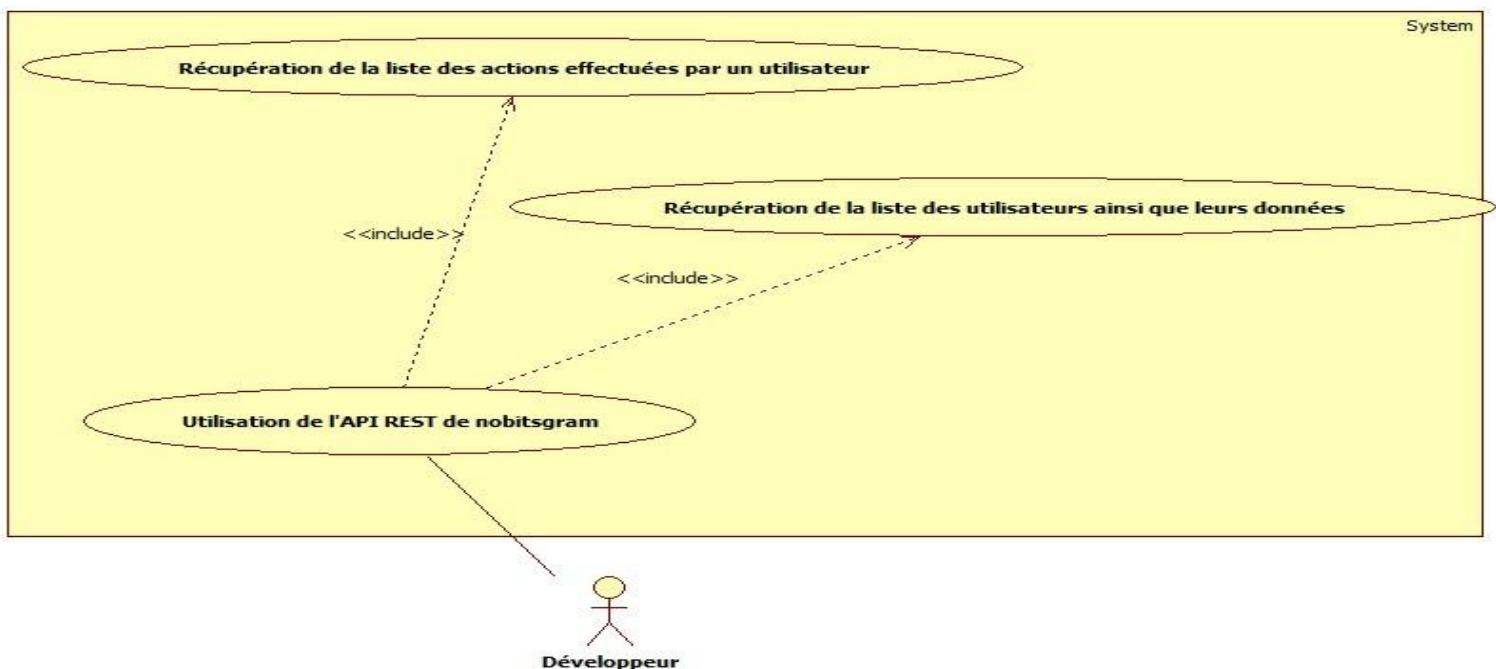
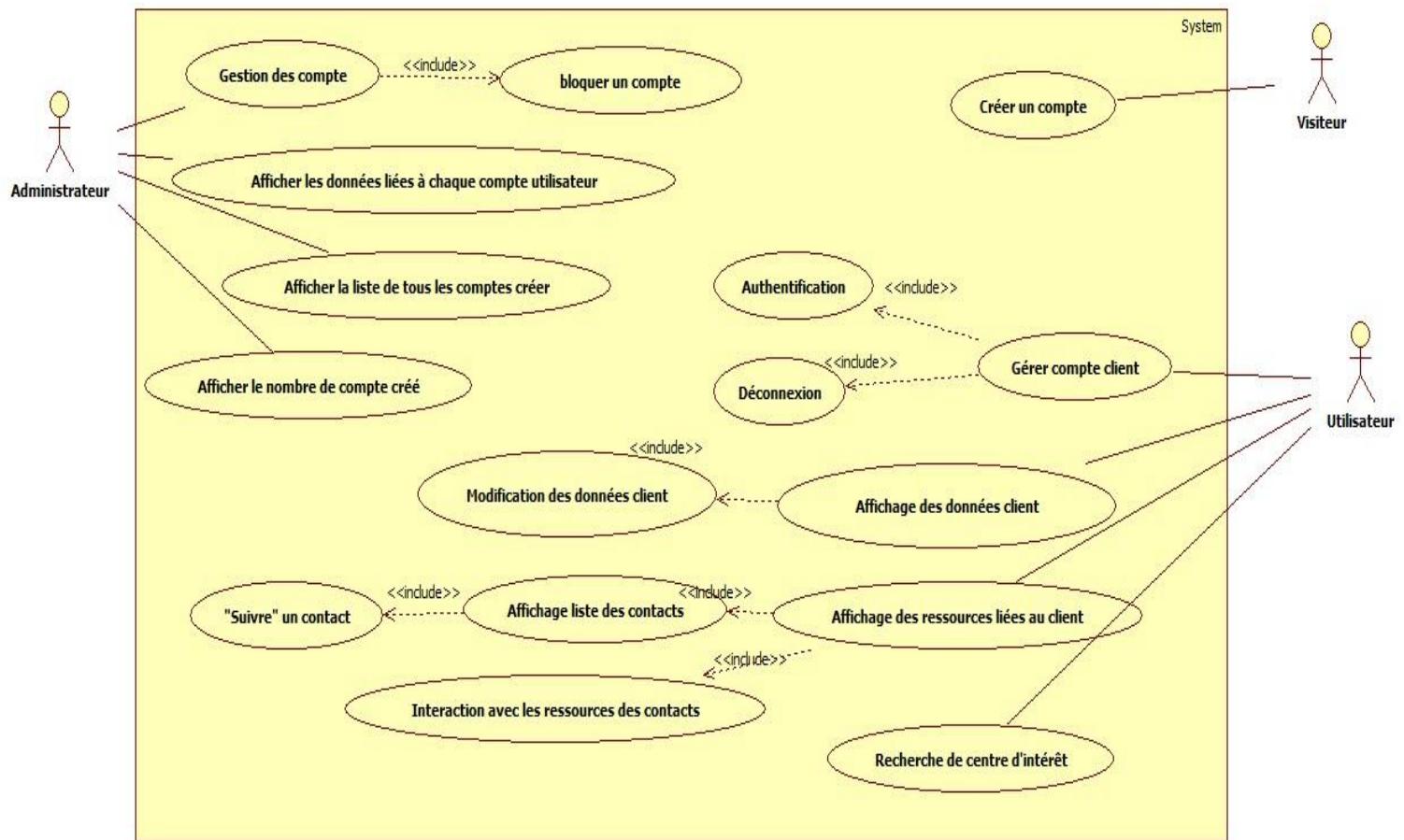


Figure 10 Use case développeur



**Figure 11 Use case 1 (Administrateur, utilisateur, visiteur)**

Les diagrammes « Use Case » ont été construits sur la base des spécifications du projet. Ainsi, dans les cas d'utilisations de l'utilisateur, nous remarquons qu'il peut gérer un compte (s'authentifier et se déconnecter). Dans ce cas d'utilisation, nous devons mettre en place une interface permettant à l'utilisateur de s'authentifier et de se déconnecter de son compte.

Un autre cas d'utilisation important est l'affichage des ressources liées au client (utilisateur). Dans ce cas précis nous parlons de ressources provenant d'Instagram, c'est-à-dire les photos taggées, les photos que le client aime « like », les « followers » et les personnes qui ont pour « follower » le client (l'utilisateur). Comme précédemment, l'application doit fournir des interfaces afin que le client puisse interagir avec ses données se trouvant sur Instagram.

Les cas d'utilisations de l'administrateur se résument à la gestion des données des clients de nobitsgram et à la gestion de leur compte. Ainsi, l'administrateur a la possibilité de bloquer un compte si le propriétaire du compte ne respecte pas les clauses du contrat de nobitsgram. Il a aussi la possibilité de voir la liste de tous les utilisateurs ainsi que toutes les actions qu'ils ont effectuées depuis la création de leur compte.

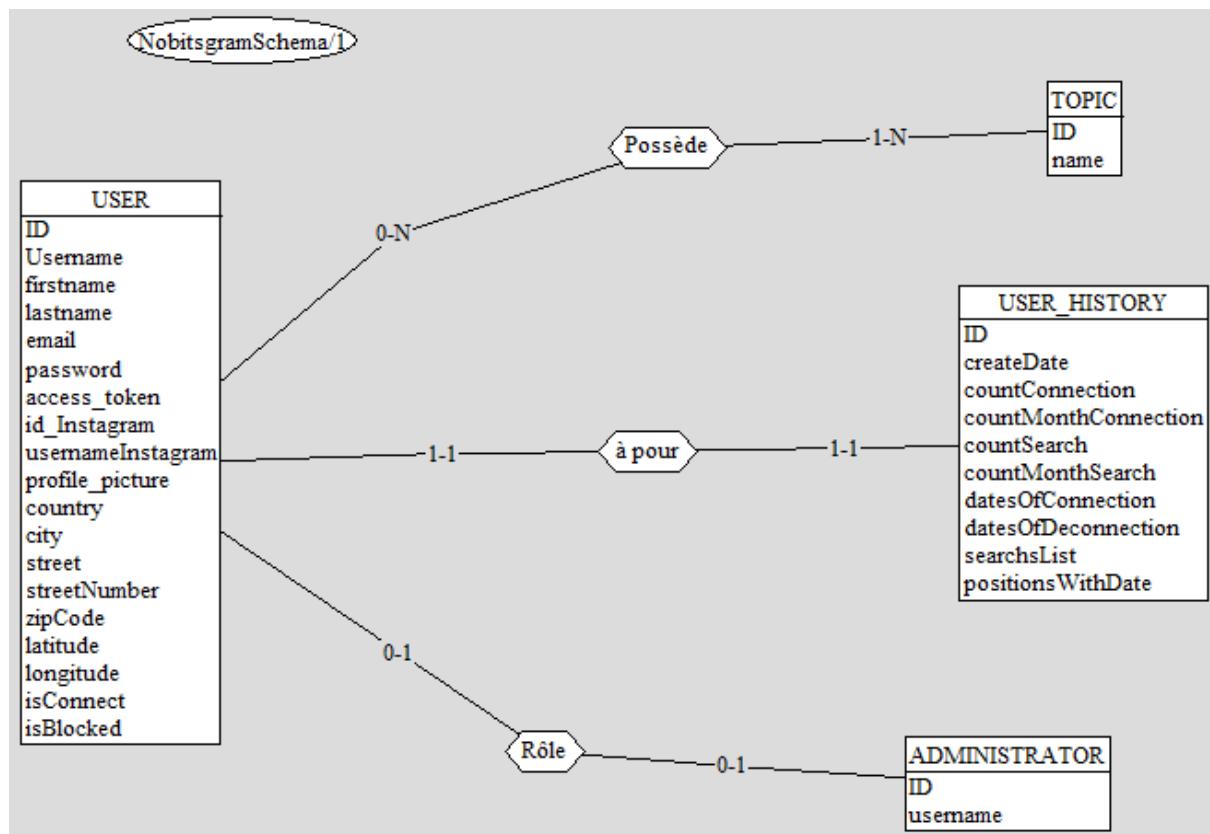
La troisième personne intervenant dans les cas d'utilisations est le développeur. Au travers d'une API REST, notre application devrait fournir des services aux développeurs voulant accéder à nos

ressources. Les services qui devront être proposés sont la récupération des utilisateurs de nobitsgram ainsi que leurs données et la liste des actions qu'ils ont effectuées.

Notre implémentation, prévoit un cas d'utilisation pour un quelconque visiteur de nobitsgram. Le visiteur est considéré comme un utilisateur anonyme et il a juste la possibilité de créer un compte sur nobitsgram, dans le cas où il n'en possède pas encore un.

Vu que nous aurons à manipuler des données, il s'avère nécessaire de penser à l'allure de notre base de données.

### 7.2- Schéma de la base de données



**Figure 12 Schéma de la base de données de nobitsgram**

La première partie de l'application consiste à mettre sur pieds une interface sur laquelle un utilisateur de nobitsgram pourra interagir avec ses données se trouvant sur le serveur d'Instagram. Un utilisateur est défini par son prénom (firstname), son nom (lastname), son pseudo (username), éventuellement son adresse (adresse converti en coordonnées géographique – longitude et latitude –), son pays (country), ainsi que quelques données (access token, ID\_Instagram, Username\_Instagram) reçues d'Instagram lors de son enregistrement sur nobitsgram.

L'utilisateur à la possibilité de définir ses centres d'intérêts (topic). Ainsi, un utilisateur peut avoir plusieurs centres d'intérêts (topics) et un centre d'intérêt pourrait être partagé par plusieurs utilisateurs. Nous aurons alors une entité pour représenter un centre d'intérêt.

La relation qui lie l'entité Utilisateur (User) à l'entité centre d'intérêt (Topic) est de cardinalité 0-N. En effet, un utilisateur peut avoir 0 ou plusieurs centres d'intérêts. Par contre, la cardinalité de

## Projet de diplôme Nobitsgram

« Topic » à « User » est de 1-N. Un topic doit nécessairement être rattaché à un User, sinon il n'a pas sa raison d'être. Un utilisateur peut avoir le rôle d'administrateur.

L'administrateur de l'application est représenté par une entité. Lorsque l'utilisateur à le rôle d'administrateur, il peut accéder à toutes les données historiques des autres utilisateurs.

Les données historiques d'un utilisateur sont représentées par une entité (User\_History). Cette entité possède une liste de date de connexions, de recherches et une liste des différents mots clés recherchés par l'utilisateur.

## 7.3 – Crédation de la base de données avec netbeans

### 7.3.2 – Crédation du projet application web « nobitsgram »

Notons qu'au cours de ce projet, nous avons travaillé avec une machine tournant sur le système d'exploitation Windows 7.

Par défaut, durant l'installation de netbeans, ce dernier met un raccourci sur le bureau. Faire un double clic sur le raccourci pour le lancer, ou s'il ne s'y trouve pas, alors lancer netbeans depuis le **menu démarrer>Tous les programmes>Netbeans>Netbeans IDE xxx** (xxx représente la version de netbeans). Après le lancement de netbeans, une page d'accueil s'affiche comme ceci :

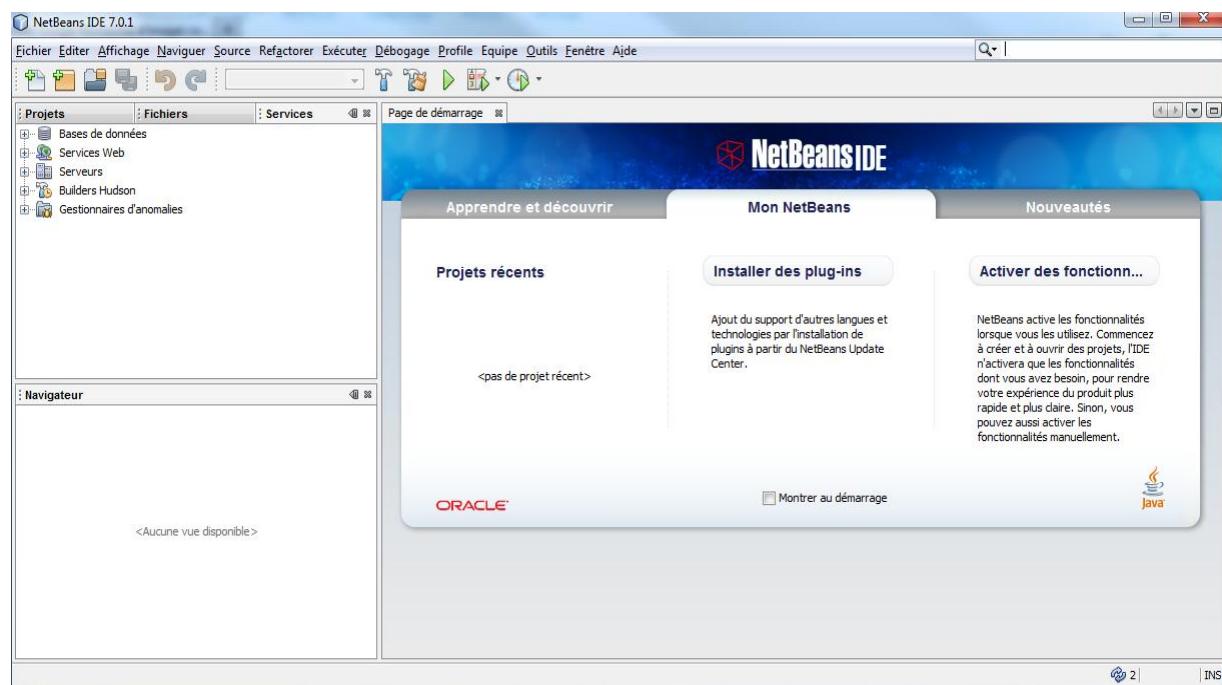


Figure 13 page d'accueil de netbeans 7.0.1

Après aller dans le menu « fichier » situé dans le coin supérieur gauche de la fenêtre de netbeans. Aller sous **fichier>Nouveau projet** comme le montre la figure ci-dessous :

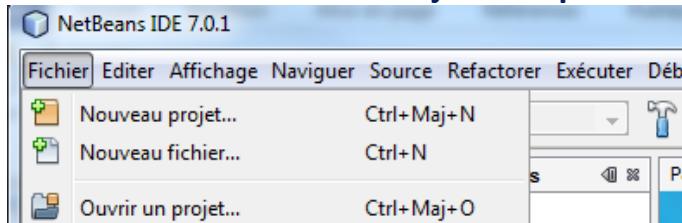


Figure 14 Création du projet nobitsgram

Ceci nous conduira à une nouvelle fenêtre dans laquelle se trouvent tous les types de projet qu'on peut créer sur netbeans. Le type de projet qui nous intéresse pour notre application est Java Web. Choisir Java Web pour la catégorie et Web Application pour le projet comme nous le montre la figure ci-dessous :

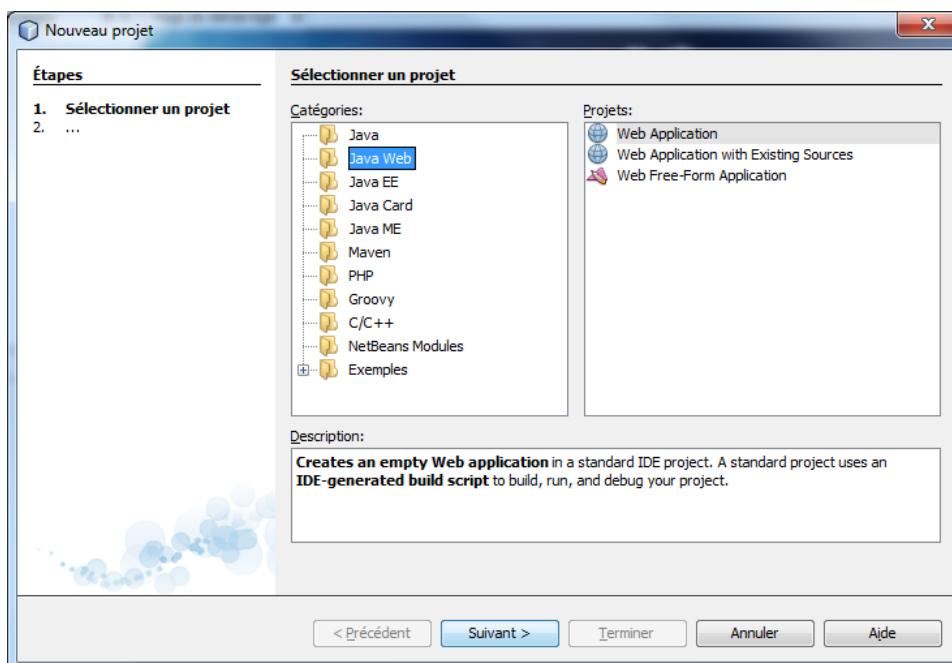


Figure 15 Sélection du type de projet de nobitsgram

Cliquer ensuite sur le bouton « suivant » pour donner le nom et lieu où le projet doit être créé. Comme nous l'avons dit plus haut, notre application a pour nom « nobitsgram ». La figure ci-dessous nous illustre cette étape.

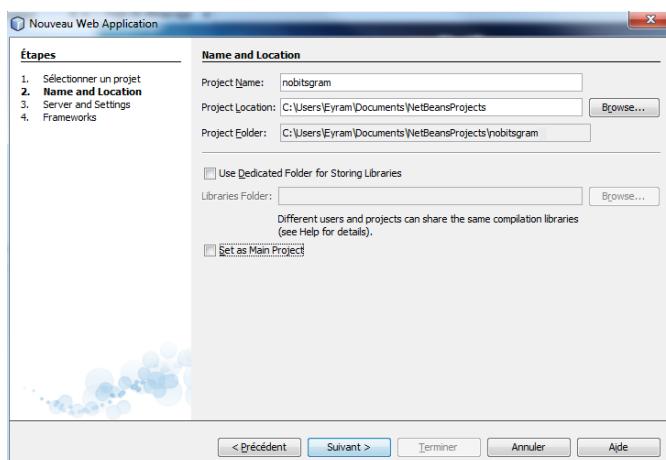
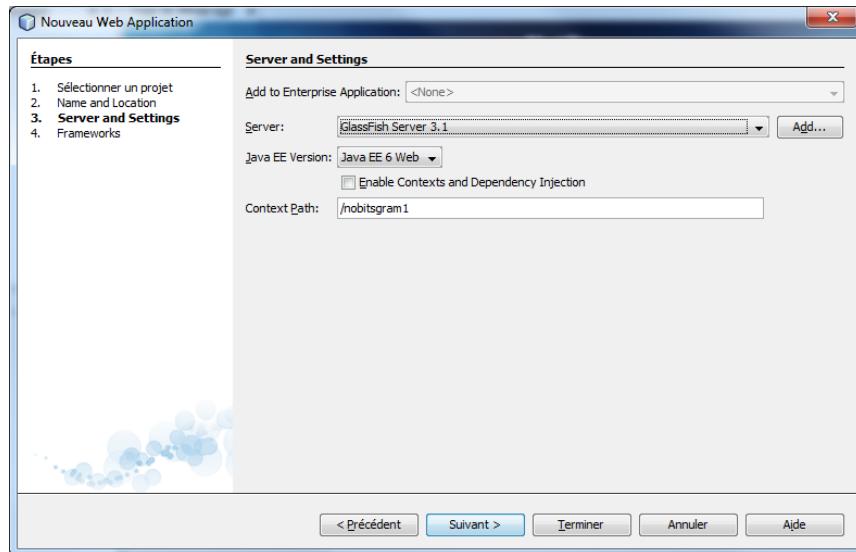


Figure 16 Nom et localisation du projet nobitsgram

## Projet de diplôme Nobitsgram

Faire encore un clic sur le bouton suivant. Cette action va nous ouvrir une nouvelle fenêtre dans laquelle nous pourrons choisir le type de serveur et la version de Java EE qu'on voudrait utiliser pour notre application. Nous choisissons le serveur Glassfish (§section 6.1.5) et la version 6 de Java EE. Notons que nous pouvons définir le path qu'on devrait utiliser pour afficher la page d'accueil de l'application. Par défaut, le path est le nom de l'application.



Nous ne choisirons pas pour le moment de frameworks, ainsi nous cliquons directement sur le bouton « terminer ». Après toutes ces étapes, notre projet nobitsgram se crée avec toutes les composantes dont il aura besoin. Le projet apparaîtra comme ceci dans l'onglet « projet » :

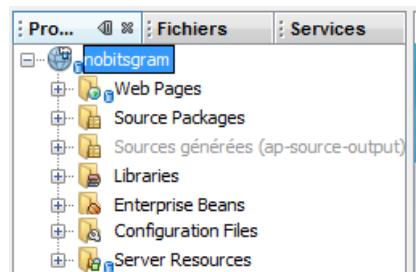


Figure 17 affichage des différentes composantes du projet nobitsgram

Nous créons ensuite six (6) package (paquetage) différents en faisant un clic droit sur le dossier **Source Package>Nouveau>Package java...** À la fin nous obtenons le résultat ci-dessous :

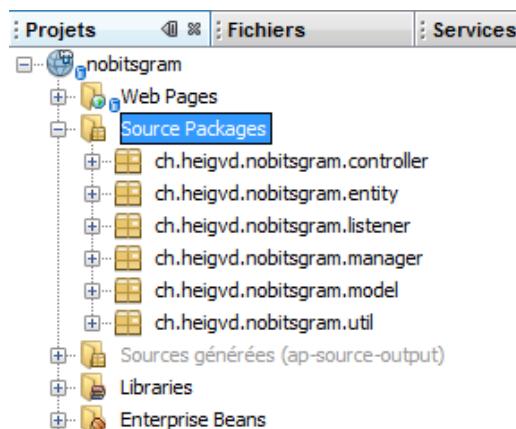


Figure 18 vue d'ensemble des packages de l'application nobitsgram

### 7.3.2 – Crédation de la base de données

Aller sous **service> Bases de données > Java DB > click droit > Créer une base de données** comme le montre la figure ci-dessous.

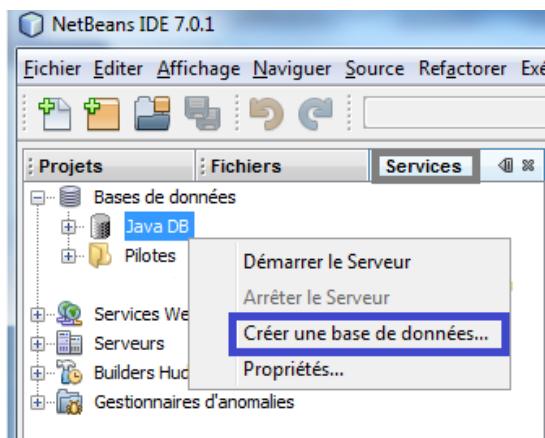


Figure 19 Image montrant le menu de service

Ensuite donner le nom de « nobitsgram\_db » à la base de données. Choisir « admin\_nobitsgram » comme nom d'utilisateur, et comme mot de passe « password\_123 ».

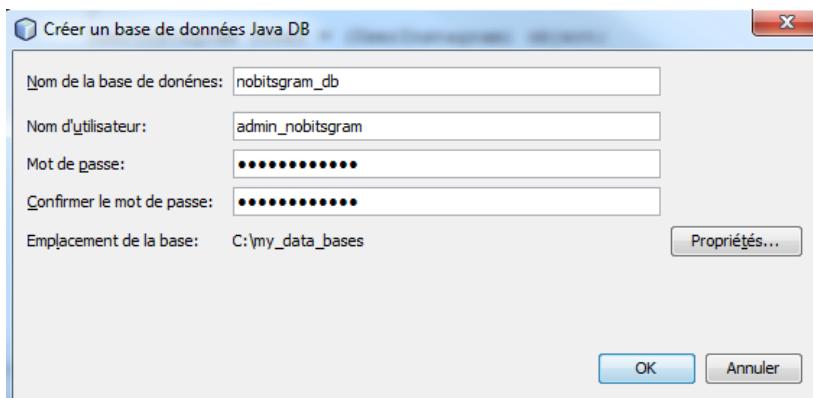


Figure 20 Crédation de la base de données nobitsgram\_db

Ensuite cliquer sur « propriétés... » pour indiquer le lieu d'installation de Java DB (par défaut, il se trouve dans le dossier de SUN). Indiquer aussi le dossier dans lequel devrait se trouver la base de données. Ici le dossier est « my\_data\_bases » et il doit être créé sur le disque C:\

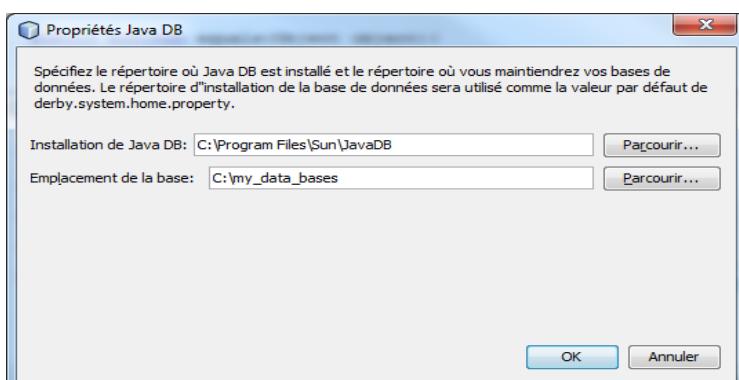
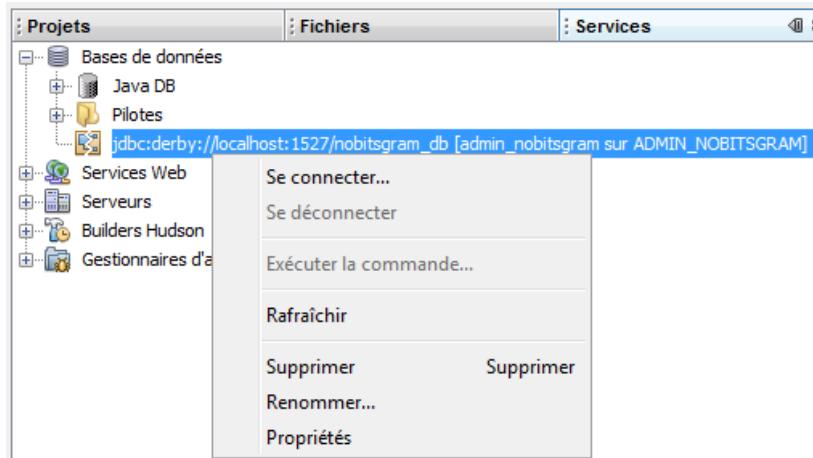


Figure 21 propriété de la base données

Ensuite Cliquer sur « OK » pour sortir de la fenêtre de propriétés Java DB et encore sur « OK » pour finalement créer la base de données en question.

Après cette opération la base de données devrait apparaître comme ci :



**Figure 22 Apparition de la base de données créée**

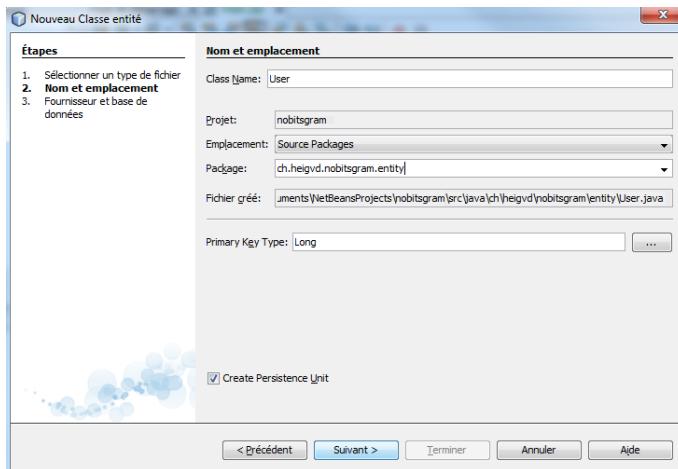
Faire un clic droit sur la base de données pour la connecter afin de voir les éléments se trouvant dans les tables.

### 7.3.3 – création du pool de connexion

La création du pool de connexion se fera durant la création de la première entité. Pour commencer, nous décidons de créer premièrement l’entité User.

Java met à disposition des outils permettant de créer des entités d’une base de données comme des classes java. La différence entre une classe java standard et une classe java entité résulte dans le fait qu’avant la déclaration de la classe il faut ajouter l’instruction « **@javax.persistence.Entity** ». Toutes les entités seront créées dans le package qui leur est dédié à savoir « **ch.heigvd.nobitsgram.entity** ».

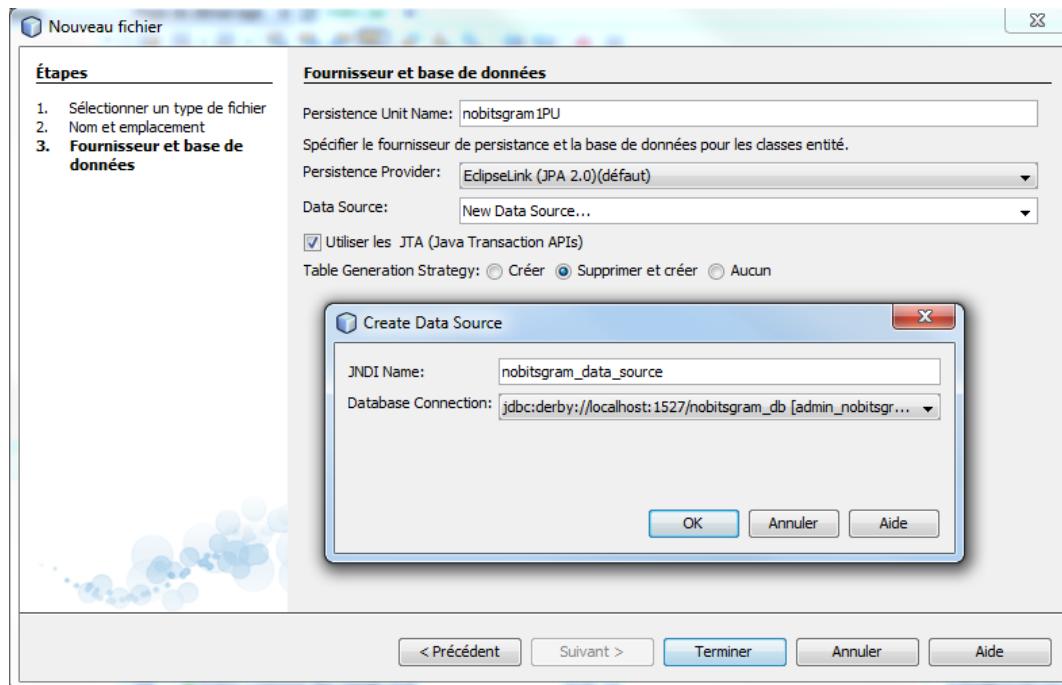
Pour créer la classe entité «User» il suffit de faire un clic droit sur le package dans lequel on souhaite qu’il soit créé. Aller sous **Nouveau>Classe entité...**. Ensuite une fenêtre comme ceci apparait :



**Figure 23 Crédation de l’entité User**

## Projet de diplôme Nobitsgram

Ensuite faire un clic sur le bouton « **Suivant >** ». Cette action nous amène sur la fenêtre où nous devons choisir le fournisseur de persistance et le nom de l'unité de persistance. Nous laissons le Persiste Unit Name à « nobitsgram1PU » et le « Persistence Provider » à « EclipseLink (JPA 2.0) ». Ensuite nous créons une nouvelle Data Source.



**Figure 24** création du pool de connexion

En cliquant sur « **New Data Source ...** » une fenêtre s'affiche. Dans cette fenêtre nous donnons comme nom JNDI le nom « **nobitsgram\_data\_source** ». Ensuite nous choisissons la base de données créée au premier point comme « **Database Connection** ».

Cliquer sur « **OK** » puis sur « **Terminer** » pour terminer l'opération. La base de données est donc connectée au projet nobitsgram.

Nous avons spécifié plus haut, que nous optons pour le modèle MVC (Modèle – Vue – Contrôleur) comme architecture pour ce projet. Il convient dès lors d'expliquer comment cette architecture sera mise en place.

## 7.4 – Modèle MVC (Modèle – Vue – Contrôleur)

### 7.4.1- Modèle

Le rôle du modèle de représenter les données du domaine et de fournir les méthodes permettant l'accès et la modification. Il est indépendant de la logique applicative. Le modèle dans notre cas sera les EJB (Enterprise Java Beans). En effet, les EJBs que nous implémenterons devront mettre à disposition des méthodes permettant le CRUD (Create Read Update Delete). Ainsi, les EJBs devront permettre de créer de nouvelles données (User et Topic – pour le moment –), de modifier leurs attributs, d'édition leurs attributs, de mettre à jour l'élément après une modification et de supprimer l'élément.

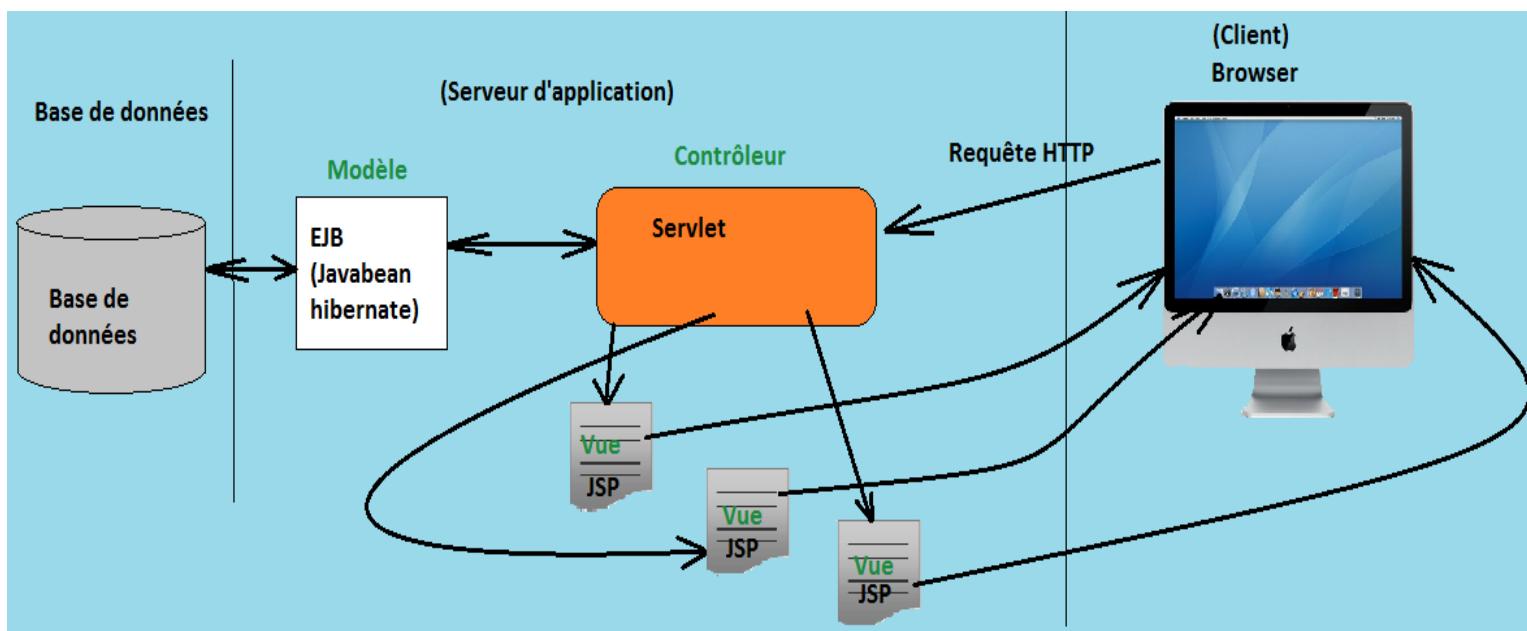
## 7.4.2- Vue

La vue est ce qui apparaît dans le « browser » de l'utilisateur. Dans notre cas, c'est du code HTML interprété par le navigateur du client. Dans notre cas, la technologie de production de contenu utilisée est la page JSP (Java Server Page). La vue utilise le langage HTML éventuellement Javascript et JQuery. Il est à noter qu'à aucun moment, la vue ne pourra modifier les données affichées. Elle joue juste le rôle de processus de fabrication de page Web en fonctions des données qui lui sont fournies.

## 7.4.3- Contrôleur

Le contrôleur est le module qui fait le lien entre le modèle et la vue. Il communique avec le modèle pour traiter des informations reçues à partir de la vue ou soit pour aller chercher les informations afin de les envoyer à la vue pour qu'elle les affiche. Le contrôleur décide de la vue à présenter en fonction des informations reçues du modèle.

Le contrôleur transforme les requêtes utilisateurs en requêtes métiers. Ainsi, il aura pour tâche de vérifier les données entrantes avant de les soumettre au modèle. La logique veut que pour chaque cas d'utilisation, nous ayons un contrôleur pour le gérer. Ainsi, nous aurons un contrôleur pour gérer la création de compte utilisateur (RegistrationServlet), un autre pour gérer la connexion (LoginServlet), encore un autre pour la modification des données du client (ServletPersonnalPage), ainsi de suite.



**Figure 25 Schéma représentant l'architecture MVC (Modèle - Vue - Contrôleur)**

L'étape suivant avant le début de l'implémentation proprement dite est l'établissement d'un diagramme UML.

## 8 – Interaction entre les différents modules de l'architecture MVC

Pour une bonne mise en œuvre de nobitsgram, il s'avère nécessaire d'établir un diagramme présentant les interactions entre les différentes parties de l'application. D'où le diagramme UML. Notons que principalement dans notre cas, nous nous sommes permis d'ajouter quelques modifications aux différents diagrammes afin de bien saisir l'interaction entre les différents modules de l'architecture MVC.

### 8.1- Page d'accueil

L'élément qui se charge de présenter la page d'accueil est le fichier « **pagelogin.jsp** ». C'est la page d'accueil de nobitsgram. Toute personne arrivant sur cette page est considérée comme utilisateur anonyme à moins qu'il se connecte sur son compte. Cette page est assez simple, et présente juste un champ pour le login et un lien vers l'adresse d'authentification OAuth d'Instagram. Outre ces deux composantes, selon la spécification, la page affiche dynamiquement et aléatoirement chaque 5 secondes, une image provenant d'Instagram et taggé avec le mot « nobits ». Cette page est attribuée comme page de démarrage à l'application Instagram. Pour afficher aléatoirement des images provenant d'Instagram, il faut au préalable aller chercher ces images sur le serveur de ce dernier. Nous avons choisi comme architecture de notre application, le modèle MVC, il est donc clair que c'est à la servlet gérant la page d'accueil de fournir les images. Ainsi, dans le code de la page « **pagelogin.jsp** » nous avons mis un contrôleur qui vérifie, si elle (**pagelogin.jsp**) a déjà reçu, de la servlet, la liste des urls des photos à afficher. En effet, dans lorsqu'une requête est envoyée à la servlet, celle-ci va interroger le serveur Instagram pour récupérer la liste des photos « tagguée » avec le mot clé « nobits ».



Figure 26 page d'accueil de nobitsgram

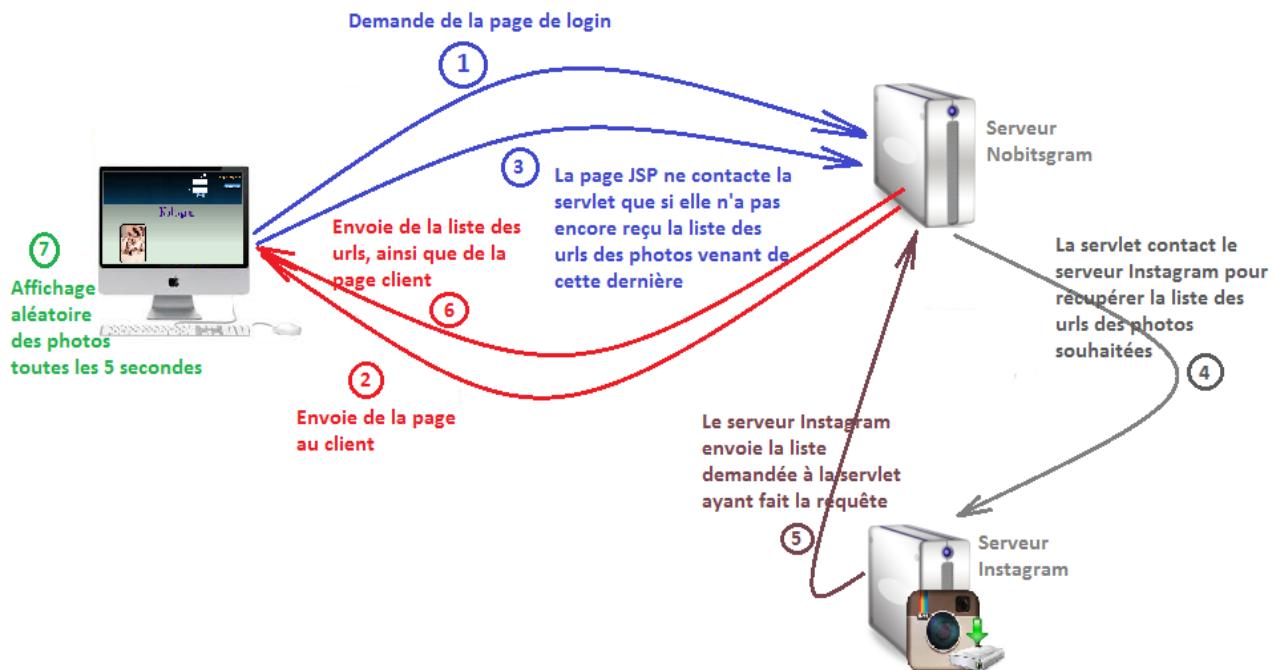


Figure 27 Diagramme montrant l'implémentation de l'affichage des photos, rafraîchies toutes les 5s

Lorsque la page « loginpage.jsp » reçoit la liste des URLs des photos, nous utilisons JavaScript pour rafraîchir dynamiquement la balise dans laquelle cette URL est appelée.

## 8.2- Enregistrement et page d'enregistrement

Avant tout enregistrement, l'utilisateur doit posséder un compte Instagram. Ainsi, lorsqu'il sera dirigé vers l'adresse d'authentification OAuth d'Instagram, il pourra recevoir le code qui permettra à nobitsgram de récupérer ses données (access token, ID, Username) sur Instagram. Lorsque l'utilisateur se sera authentifié sur Instagram, il sera redirigé vers la servlet « RegistrationServlet » qui s'occupera de recevoir le code et de communiquer avec le serveur d'Instagram afin de recevoir les données du client. Ensuite la servlet redirigera le client vers la page JSP d'enregistrement sur laquelle se trouve un formulaire. Pour s'enregistrer, l'utilisateur clique sur le bouton ou le lien « register ». Ensuite il est redirigé sur la page d'authentification d'Instagram.



The screenshot shows the Instagram login page. The URL in the address bar is [https://instagram.com/accounts/login/?next=/oauth/authorize%3Fclient\\_id%3D5e2a174a3980461984C](https://instagram.com/accounts/login/?next=/oauth/authorize%3Fclient_id%3D5e2a174a3980461984C). The page features the Instagram logo at the top. Below it, there are two input fields: 'Nom d'utilisateur:' (Username) and 'Mot de passe:' (Password). A 'Connexion' (Login) button is centered below the password field. To the right of the password field, there is a link 'Mot de passe oublié ?' (Forgot password?). The browser's toolbar is visible at the top, showing various icons for navigation and settings.

Figure 28 page d'authentification d'Instagram

## Projet de diplôme Nobitsgram

Après que l'utilisateur ait se soit authentifié auprès d'Instagram, une page d'autorisation apparaît demandant à l'utilisateur s'il souhaite que ses données Instagram puissent être utilisées par l'application cliente Nobitsgram.

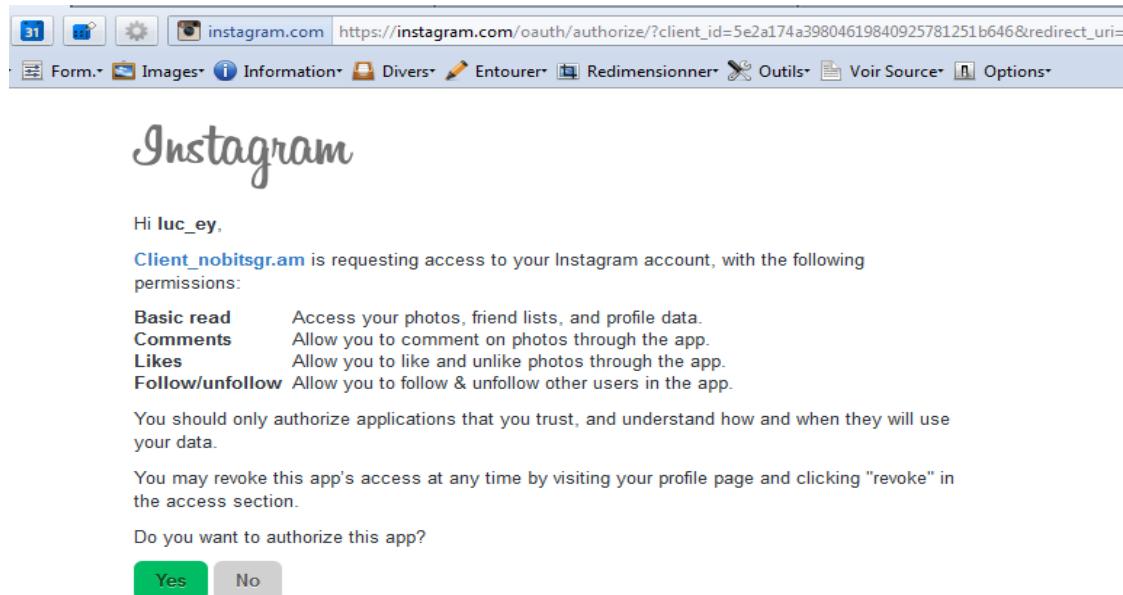


Figure 29 page d'autorisation d'Instagram

Lorsque l'utilisateur accorde l'autorisation à nobitsgram il redirige vers la page d'enregistrement de nobitsgram.



The screenshot shows a web browser window with the URL <http://localhost:8080/nobitsgram/RegistrationServlet?code=c32bf88dfa0f4029b84b533f02c34e06>. The page title is "NOBITSGRAM". The main heading is "REGISTRATION". There are five input fields for "First name", "Last name", "Street number", "Street address", and "city".

Figure 30 page d'enregistrement de nobitsgram

## 8.3- Servlet gérant l'enregistrement (RegistrationServlet)

La réception des données venant du serveur Instagram se fait au travers de la méthode GET. Ces données seront « parsés » afin d'en extraire l'access\_token, le username Instagram de l'utilisateur, l'ID Instagram de l'utilisateur et l'url sur menant à sa photo de profile.

Puisque nous utilisons des servlets et que ses paramètres globaux sont partagés entre toutes les utilisateurs qui y font une requête, il est nécessaire de trouver le moyen d'envoyer les données reçus au travers de la méthode GET à la méthode Post, méthode qui se charge de valider le formulaire d'enregistrement et d'enregistrer un utilisateur dans la base de données de nobitsgram. Tout utilisateur qui accède à la page d'accueil de nobitsgram se vera attribué une session unique et qui durera jusqu'à ce que l'utilisateur la détruise en fermant son navigateur (si le navigateur n'enregistre pas les cookies) ou jusqu'à ce la durée de cette session expire. Pour le cas de notre application, nous avons mis la durée de la session à 7 minutes. Ainsi, après un temps d'inactivité de 7 minutes, la session expire et si l'utilisateur s'il veut continuer par interagir avec le site se vera attribué une autre session.

Donc, la servlet envoie les données (access\_token, username Instagram, Id Instagram, url de la photo de profile Instagram) à la session de l'utilisateur. Quel que soit la page sur laquelle l'utilisateur se trouve, ces données sont accessibles tant qu'ils n'ont pas été détruits. Ainsi, lorsqu'au travers de la méthode POST, la servlet recevra les données du formulaire, elle pourra au même moment récupérer les données qu'elle avait mises préalablement dans la session. Lorsque la servlet valide le formulaire, elle fait une redirection vers la servlet « GalleryServlet ». Cette servlet se chargera d'envoyer la page « gallery.jsp » au browser de l'utilisateur.

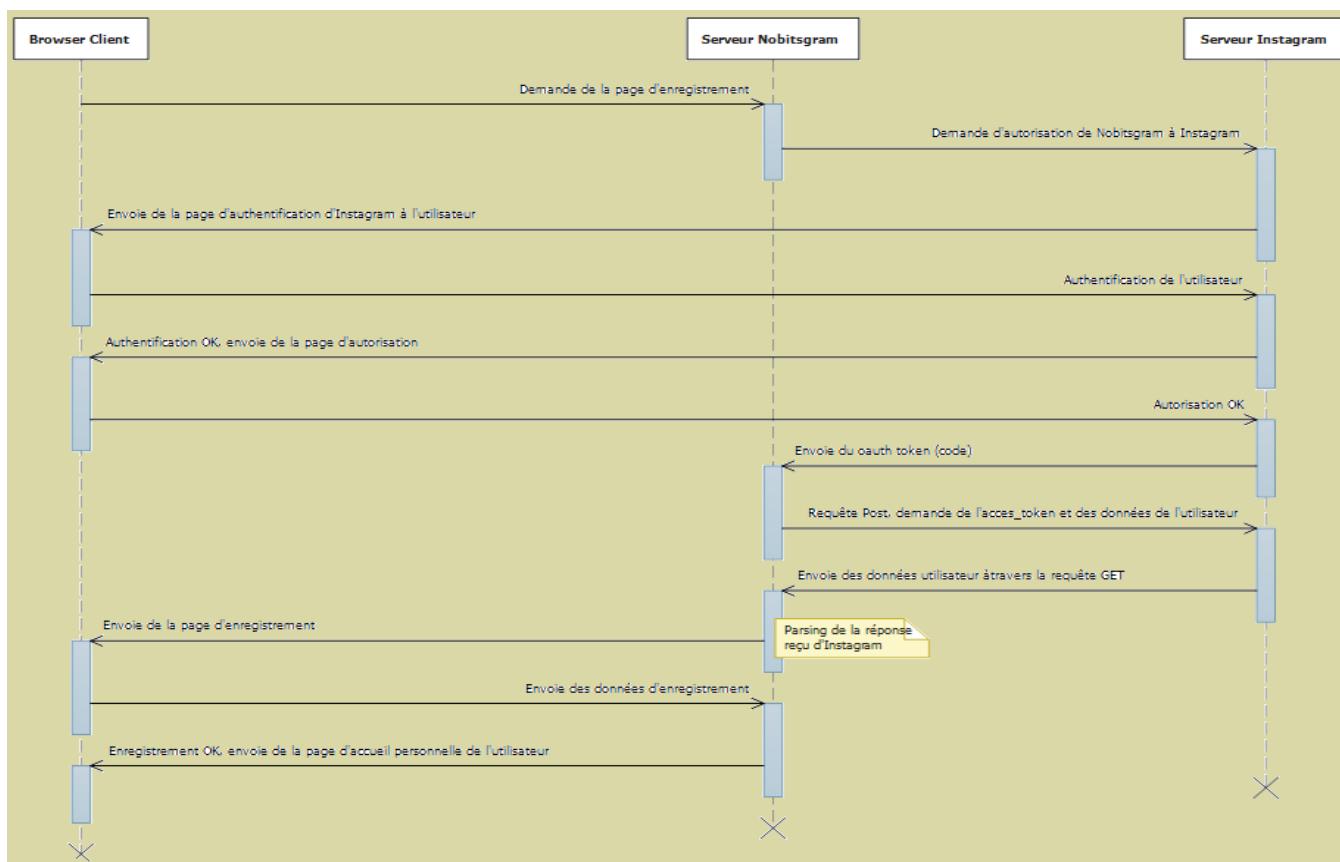


Figure 31 Diagramme de séquence montrant les différentes étapes lors de l'enregistrement d'un utilisateur

L'enregistrement se fait suivant les étapes suivantes :

➤ **Etape 1 : Réception du code à partir d'Instagram**

Lorsque le client a pu s'authentifier sur Instagram, le serveur de ce dernier envoie une requête GET au serveur de nobitsgram et dans laquelle le code est enveloppé dans le quering. Il suffit donc de faire appel à la méthode « `request.getQuering()` » pour récupérer la chaîne de caractère du code et faire un parsing pour l'y extraire. Le parsing se fait à l'aide d'une méthode static de la classe « `MyParser` ». Notons que la réception du code ne peut se faire que dans la méthode `doGet(HttpServletRequest , HttpServletResponse )`. En effet, puisque le serveur d'Instagram fait une requête « `GET` » en direction de notre serveur, le seul moyen de voir un flux de données c'est se trouver dans la méthode invoquée.

➤ **Etapes 2, 3 et 6': Redirection vers la page d'enregistrement**

La particularité d'une servlet est de partager ses attributs globaux avec toutes les instances qui y font appel. L'idée de déclarer une variable globale et ensuite de l'affecter au code pour le réutiliser comprend le risque que le code soit le même pour tous les utilisateurs. Ce qui est absurde. Une façon d'y remédier est d'envoyer le d'assimiler le code à la session, à travers la méthode « `request.getSession.setAttribute( "code", code ) ;` », créer par celui qui s'enregistre. Ainsi, lorsqu'il aura fini de remplir le formulaire et qu'il le soumettra à la servlet, cette dernière va récupérer le code à la même occasion que les valeurs des champs qu'il aura rempli.

➤ **Etape 4 : Communication avec le serveur Instagram**

La communication avec le serveur Instagram se fait par l'intermédiaire de la requête Post. Le serveur de nobitsgram envoie une requête Post dans laquelle il aura ajouté les paramètres de requêtes suivants :

```
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)
```

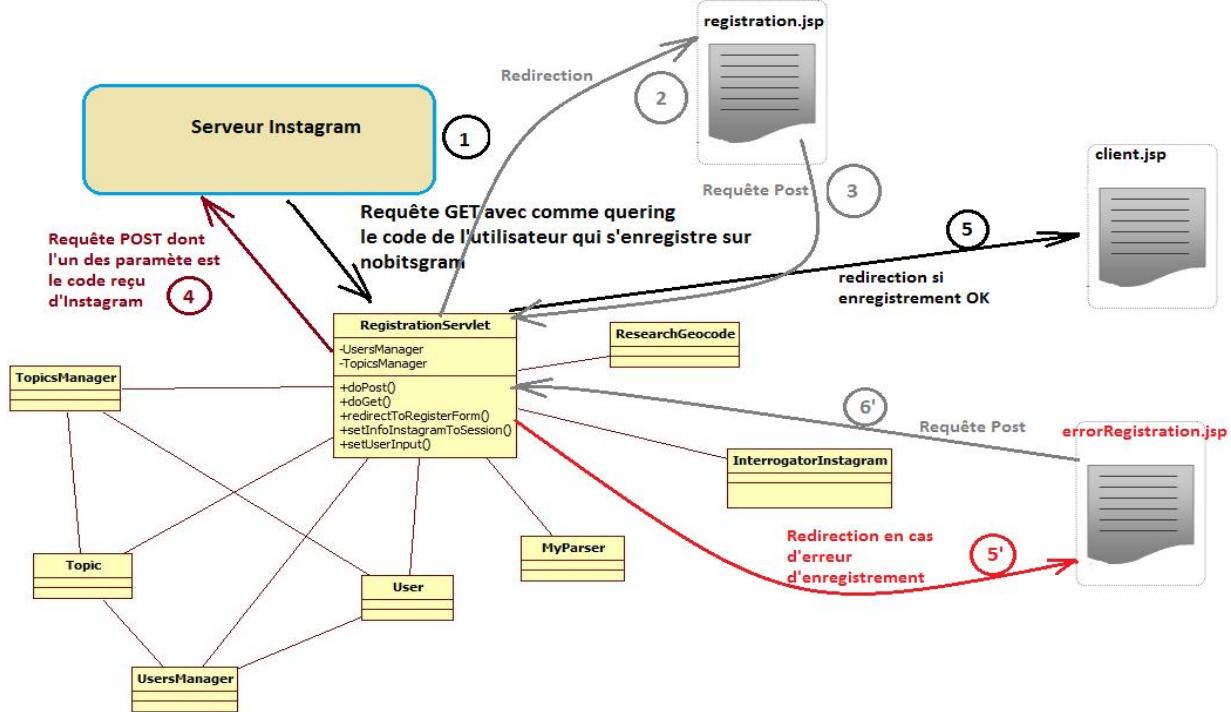
Si le serveur accepte la requête, il renverra une réponse au travers d'une méthode Post, réponse dans laquelle se trouveront les informations (access token, ID, Username,...) concernant le compte Instagram du client.

➤ **Etape 5' : Redirection vers la page d'erreur d'Instagram**

Lorsqu'une erreur est trouvée dans les valeurs des champs soumis à la servlet, l'utilisateur est redirigé vers la page d'erreur du formulaire.

➤ **Etape 6 : Redirection vers la page client**

Si le formulaire soumis par le client qui s'enregistre est accepté, le client sera automatiquement redirigé vers sa page client personnel. Sur cette page, le client pourra interagir avec ses données récupérées sur le serveur d'Instagram et avec les ressources se trouvant sur le dit serveur.

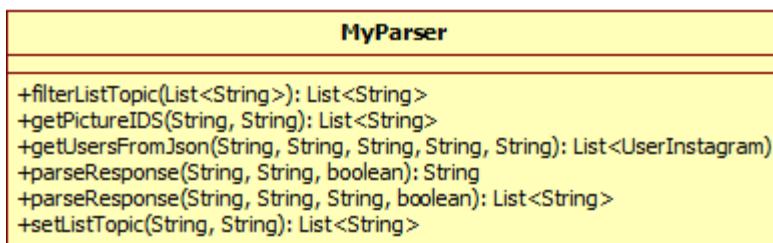


**Figure 32 Diagramme UML de la Servlet gérant l'enregistrement et interactions avec les pages JSP correspondantes**

Durant l'enregistrement, la servlet RegistrationServlet a utilisé les services de la classe MyParser afin d'extraire les données souhaitées. En effet, toute réponse venant d'Instagram est sous forme de fichier JSON. Chaque réponse reçue contient des certaines données dont nous n'avons pas besoin. Il est donc nécessaire de pouvoir extraire les données voulues de chaque réponse données. D'où l'implémentation de la classe MyParser.

## 8.4 – La classe MyParser

Cette classe se trouve dans le package « `ch.heigvd.nobitsgram.util` ». Elle a pour rôle de « parser » les fichiers de type JSON (§ partie 6.1.8). MyParser est static, ce qui permet une utilisation de ses méthodes indépendamment de son instanciation.



MyParser met à disposition plusieurs méthodes intéressantes. Au nombre de celles-ci nous pouvons retenir :

❖ **`filterListTopic(List<String> ) :List<String>`**

Cette méthode est prend en paramètre une liste de chaîne de caractère, supprime tous les doublons et retourne la liste ainsi filtrée.

## Projet de diplôme Nobitsgram

### ❖ **getPictureIDS(String message, String path) :List<String>**

Cette méthode prend en paramètre le contenu du fichier qu'on veut parser (premier paramètre), et le nom de l'élément qu'on veut extraire (deuxième paramètre). Cette méthode a été implémentée car dans le fichier JSON contenant les informations sur un utilisateur Instagram, il y a deux types de nom « id » et tous les deux ne veulent pas dire la même chose. Le premier définit les « id » des utilisateurs, et le second celui des photos. Ils sont tous mélangés et la seule différence réside dans le format de l' « id ». En effet, l' « id » d'un utilisateur est une succession de chiffres (exemple 10840565), alors que celui d'une photo est formé par deux blocs de succession de lettre et séparé par le caractère « \_ » (exemple 10840565\_7892133). Le premier bloc défini l'« id » du propriétaire de la photo et le second celui de la photo. Ainsi, chaque photo est rattachée à son utilisateur.

Après la récupération de la liste de tous les « id » se trouvant dans le contenu du fichier JSON, on supprime de la liste tous ceux qui n'ont pas un format de deux blocs. La liste ainsi filtrée est ensuite retournée.

### ❖ **getUsersFromJson(String, String, String, String, String) :List<UserInstagram>**

Le premier paramètre est le contenu du fichier JSON qu'on veut parser; le second paramètre est la racine du fichier et les autres paramètres sont les différents noms des éléments qu'on veut extraire du contenu du fichier à savoir : le « username Instagram », l'url où se trouve la photo de profile de l'utilisateur et son « id ». Ainsi, pour chaque utilisateur défini dans le fichier, un utilisateur Instagram (UserInstagram) est créé et insérée dans une liste qui sera retournée à la fin de l'exécution de la méthode.

### ❖ **parseResponse(String, String, boolean) :String**

Cette méthode prend comme premier paramètre le contenu du fichier JSON à parser. Le deuxième paramètre est le nom de l'élément qu'on veut extraire. Le troisième paramètre indique s'il est nécessaire d'enlever les guillemets (") ou pas. Cela est nécessaire dans le cas où l'élément extrait contient d'autres éléments imbriqués.

### ❖ **parseResponse(String, String, String, boolean) : List<String>**

Le premier paramètre défini le contenu JSON à parser, le second défini la racine des éléments du fichier, le troisième le nom de l'élément qu'on veut extraire et le boolean indique si l'on veut ou non enlever les guillemets (""). Cette méthode parcourt tous les compartiments du fichier pour en extraire une liste d'élément dont on a précisé le nom par l'intermédiaire du troisième paramètre de la méthode. Cette méthode retourne la liste ainsi obtenue à la fin de son exécution.

Les fichiers JSON proviennent de l'échange de données entre le serveur de nobitsgram et celui d'Instagram. Nous avons implémenté une classe (InterrogatorInstagram) qui permet de communiquer avec le serveur d'Instagram.

## 8.5 – La classe InterrogatorInstagram

La classe InterrogatorInstagram se trouve dans le package « **ch.heigvd.nobitsgram.util** ». Elle a pour rôle de communiquer avec le serveur d’Instagram. Elle permet ainsi de recevoir les réponses lors d’une requête de type GET ou POST vers le serveur d’Instagram.

<b>InterrogatorInstagram</b>
+getClientInformation(String, List<String>, List<String>): String
+getResponseOfInstagram(URLConnection): String
+getSearchResult(String): String

Parmi les méthodes intéressantes de la classe nous avons :

❖ **getClientInformation(String, List<String>, List<String>): String**

Cette méthode est celle qui demande les informations dans laquelle se trouve l’access\_token. Le premier paramètre de la méthode est l’url d’autorisation d’Instagram. Le second paramètre est la liste des paramètres à ajouter à la requête Post. Le second paramètre est la liste des valeurs des paramètres de la requête Post. La réponse reçue du serveur est retournée à la fin de l’exécution de la méthode.

❖ **getResponseOfInstagram(URLConnection): String**

L’url de toute requête est utilisé pour instancier un objet de type URLConnection. Cette méthode ouvre une connexion afin de recevoir le flux d’entrée venant du serveur Instagram. Ce flux sera transformé ensuite en chaîne de caractère et retournée à la fin de l’exécution de la méthode. Cette méthode est utilisée par getClientInformation() et getSearchResult().

❖ **getSearchResult(String): String**

Cette méthode prend en paramètre le l’URL du serveur d’Instagram et retourne la réponse de ce dernier. Elle utilise la méthode getResponseOfInstagram() pour communiquer et recevoir la réponse venant de Instagram.

## 8.6 – Classe LoginServlet

Tout utilisateur possédant un compte nobitsgram à la possibilité de se « logguer » sur son compte. La servlet qui gère le login est LoginServlet. Sur la page d’accueil de nobitsgram, un champ est prévu pour que l’utilisateur puisse se logguer. Il lui suffit de saisir son username (celui qu’il a donné lors de son enregistrement) et son mot de passe.

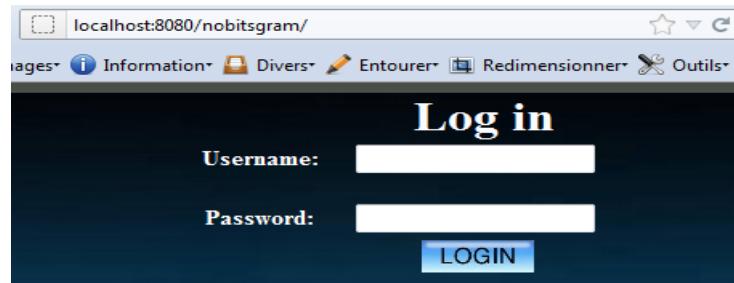


Figure 33 Champ de login de la page nobitsgram

## Projet de diplôme Nobitsgram

La JSP (pagelogin.jsp) lorsque le bouton « LOGIN » est cliqué, contacte LoginServlet avec les données que l'utilisateur à saisie. LoginServlet, va tout d'abord tenter de récupérer l'objet « User » ayant comme username celui saisi. S'il ne le trouve pas il redirige directement l'utilisateur vers une page d'erreur de login. Dans le cas où il trouve le username, il va faire un test pour voir si le mot de passe saisi et celui se trouvant dans la base de données sont identiques. Dans le cas où ils sont différents, il redirige l'utilisateur vers la page d'erreur de login. La page d'erreur de login est aussi contrôlée par la servlet LoginServlet. À chaque fois que LoginServlet n'aura pas authentié un utilisateur cette page sera retournée.



Figure 34 page d'erreur login de nobitsgram

Lorsque l'authentification de l'utilisateur a été établie, l'attribut « isConnected » indiquant si un utilisateur est connecté ou pas est mis à « true », ainsi cet utilisateur sera vu par tous les autres utilisateurs connectés comme étant connecté. Ensuite ce dernier est dirigé vers sa page personnelle.

## 8.6 – Page personnelle

La page personnelle est définie par une ensemble d'onglet se trouvant dans le coin supérieur gauche de la fenêtre.



Figure 35 Onglets de la page personnelle



Figure 36 sous onglets de l'onglet "Friends"

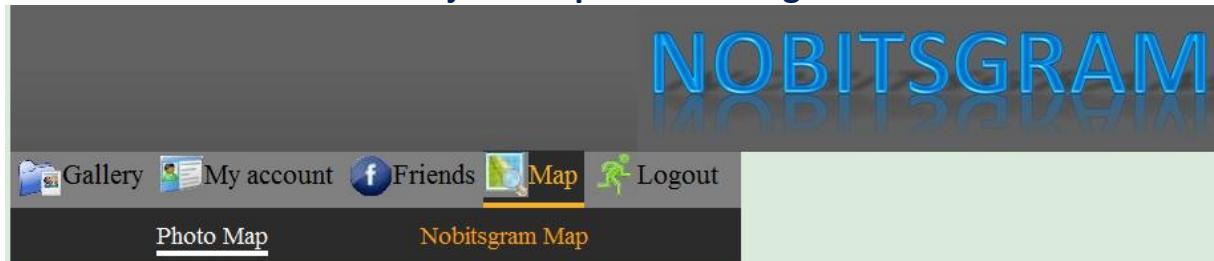


Figure 37 sous onglets de l'onglet "Map"



Figure 38 sous onglets de l'onglet "My account"

## 8.7 – Page gallery

La page gallery affiche une grille de 3X3 photos. Les 3 premières photos sont des photos choisie au hasard et parmi celles d'Instagram et taggées avec un des centres d'intérêts de l'utilisateur. Les photos de la deuxième ligne sont celles des « followers » de l'utilisateur. Et les trois dernières sont celles que l'utilisateur a aimée (« likées »).



Figure 39 page gallery

La page gallery est contrôlée par la servlet GalleryServlet.

## 8.8 – La classe GalleryServlet

Avant d'afficher les différentes photos, gallery.jsp doit recevoir de GalleryServlet trois listes contenant chacune les urls des différents types de photos à afficher. La liste d'url pour les photos de topics est obtenue en envoyant une requête de type GET au serveur d'Instagram. Dans l'url de la requête on introduit un centre d'intérêt pris au hasard dans la liste des centres d'intérêt de l'utilisateur. Ainsi, Instagram va nous retourner les photos étant taggées par ce centre d'intérêt. La réponse d'Instagram sera parsée à l'aide de la méthode statique « parseResponse() » de la classe « MyParser » afin d'obtenir une liste d'url de photo. Cette liste sera insérer dans la session de l'utilisateur et récupérée ensuite par gallery.jsp.

La création de la liste des urls des photos des followers se fait comme suit :

- Premièrement il faut récupérer la liste des « id » des followers de l'utilisateur.
- Deuxièmement choisir au hasard un « id » de followers.
- Troisièmement récupérer la liste des photos récentes postées par l'utilisateur possédant cet « id », liste qui sera insérer dans la session de l'utilisateur et récupérée ensuite par gallery.jsp

La création de la liste des urls des photos que l'utilisateur à « likées » se fait en envoyant une requête à Instagram avec comme url :

[https://api.instagram.com/v1/users/self/media/liked?access\\_token=access\\_token](https://api.instagram.com/v1/users/self/media/liked?access_token=access_token)

Le résultat reçu d'Instagram va être aussi parsée afin d'obtenir une liste qui sera à son tour insérée dans la session de l'utilisateur. Cette liste va à son tour être récupérée par gallery.jsp.

Cette page gère l'affichage de la page gallery. Elle reçoit trois listes d'url de photo venant de GalleryServlet. Pour chaque liste, elle choisit au hasard trois (3) url différents et affiche les photos correspondantes.

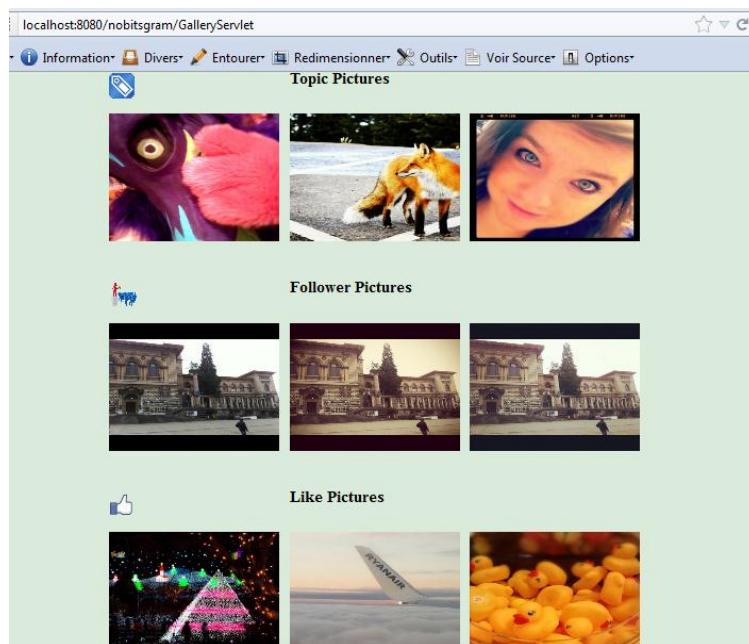


Figure 40 grille de 3X3 images de la page gallery

## 8.9- Page de recherche

Le résultat de la recherche des photos liées à un centre d'intérêt s'affiche dans la page searchTopic.



Figure 41 affichage des résultats de la recherche du mot clé "snow"



Figure 42 affichage du message d'erreur lié à la recherche du mot "Eyram"

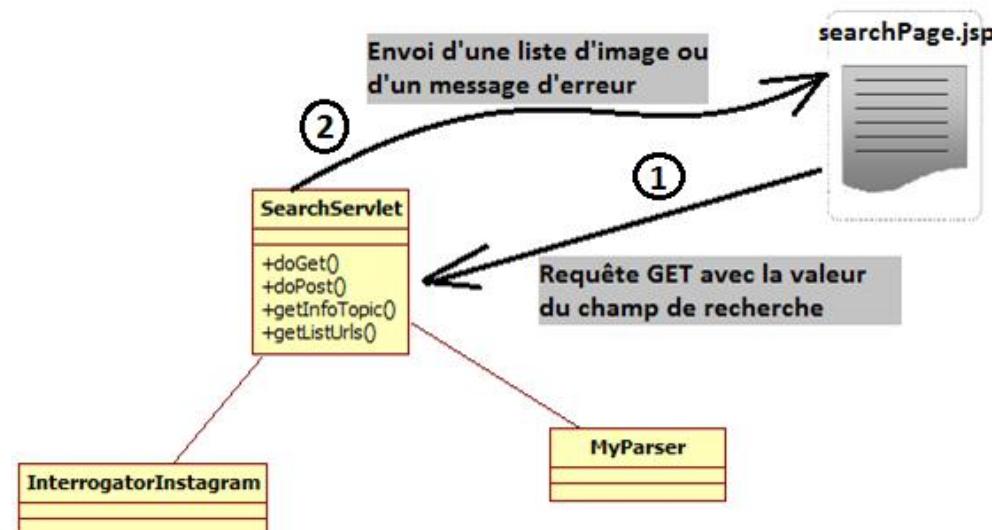


Figure 43 Diagramme UML de la servlet gérant la recherche et de son interaction avec sa page JSP correspondante

## 8.10 - classe SearchServlet

Lors d'une recherche, la page JPS sur laquelle la recherche est effectuée envoie une requête GET à la servlet, requête dans laquelle se trouve la valeur du champ de recherche. Si le champ est vide, la servlet va automatiquement envoyer un message d'erreur à sa page JSP. La page JSP, va consulter les éléments se trouvant dans la réponse de la servlet. Si elle trouve que l'élément « error » n'est pas « null » alors elle va afficher le message d'erreur envoyé par la page JSP.

Dans le cas où la valeur du champ de saisie n'est pas nulle, la servlet va construire l'url de recherche selon la documentation de l'API d'Instagram à savoir insérer le terme de la requête ainsi que l'access token de l'utilisateur selon le format suivant :

```
https://api.instagram.com/v1/tags/Terme_recherché/media/recent?access_token=ACCESS-TOKEN
```

Le serveur va envoyer une réponse dans laquelle se trouveront les liens vers les images ayant un rapport avec le terme recherché. La communication avec le serveur d'Instagram se fait au travers d'une instance de la classe **InterrogatorInstagram**. Cette instance reçoit une réponse sous forme de chaîne de caractère qu'il faudra parser afin d'en extraire les liens des images. Ces liens seront enregistrés dans une liste qui sera envoyée à la page JSP « searchPage » afin qu'elle puisse les afficher.

Si la recherche s'avère infructueuse du côté d'Instagram, son serveur retourne juste une chaîne de caractère dans laquelle ne se trouve aucun lien.

Pour connaître les informations concernant le terme recherché en l'occurrence le nombre d'image ayant un rapport avec le dit thème, il suffit d'envoyer une requête au serveur d'Instagram selon le format suivant :

```
https://api.instagram.com/v1/Terme_recherché/nofilter?access_token=ACCESS-TOKEN
```

Le serveur retournera une réponse qui aura le format suivant :

```
{  
    "data": {  
        "media_count": 472,  
        "name": "nofilter",  
    }  
}
```

Il suffit donc de parser la réponse reçue du serveur pour en extraire la donnée voulue à savoir le nombre de média (images) ayant un rapport avec le terme recherché.

## 8.11- Déconnexion d'un utilisateur

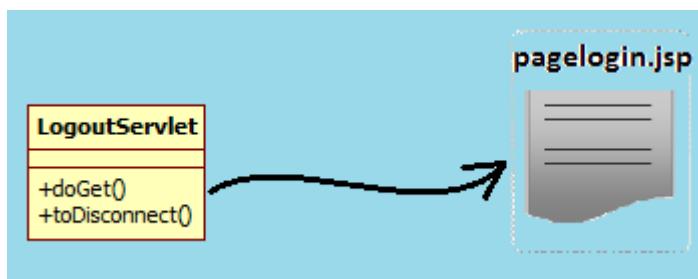


Figure 44 Diagramme de la servlet gérant la déconnexion et son interaction avec la page JSP correspondante

Lorsqu'un client se déconnecte, il effectue une requête GET à la servlet LogoutServlet. La méthode doGet() invoque la méthode toDisconnect() pour fermer la session. Avant la fermeture de la session, l'attribut « isConnected » indiquant si un utilisateur est connecté ou pas est mis à « false » afin d'indiquer aux autres utilisateurs que l'utilisateur est déconnecté. La fermeture de la session se fait au travers la méthode « `request.getSession.invalidate();` ». Après l'exécution de cette instruction, le client est redirigé vers la page d'accueil de l'application.

## 8.12 – Onglet « My Account »

### 8.12.1 – Page settingAccount

Pour modifier ses données, l'utilisateur doit aller sur l'onglet « My account ». Arrivé sur cette page il a la possibilité de voir toutes ses données et de les modifier. Les seules données qu'il ne peut pas modifier est son Identifiant (nombre) nobitsgram, son pseudo nobitsgram et ses données Instagram (access token, username, id).

The screenshot shows a web application interface. At the top, there is a navigation bar with links: "Gallery", "My account" (which is highlighted in yellow), "Friends", "Map", and "Logout". Below the navigation bar, there are two tabs: "Setting Account" (selected) and "My Medias". The main content area has a blue header with the text "Haddock, you can set your nobitsgram data here!". Below this, there is a form with four input fields:

Last name:	<input type="text"/>
First name:	<input type="text"/>
Street number:	<input type="text"/>
Street address:	<input type="text"/>

Figure 45 page settingAccount

## Projet de diplôme Nobitsgram

L'utilisateur à la possibilité d'ajouter ou de supprimer un centre d'intérêt. Lorsque l'utilisateur demande à afficher sa page personnelle, il fait une requête GET à la servlet « `ServletPersonnalPage` ». Cette servlet récupère les données du client et les envois à la page JSP « `settingAccount.jsp` ». La servlet insère la liste des topics dans la requête envoyée à la page « `settingAccount.jsp` ». La page JSP extrait les données se trouvant dans la requête et les affiche. Lorsque, l'utilisateur modifie une ou plusieurs de ses données, il envoie une requête Post à la servlet ainsi que la valeur des champs à modifier. Si aucun champ n'est modifié, la servlet valide le formulaire. Par contre si l'utilisateur modifie une ou plusieurs données et que l'une au moins d'entre elles n'est pas correcte, la servlet redirige le client vers la page « `My account` » avec un message d'erreur correspondant à l'erreur trouvée.

Dans le cas où le formulaire est accepté par la servlet et que la modification a été effectuée, le client est automatiquement redirigé vers la page `gallery`.

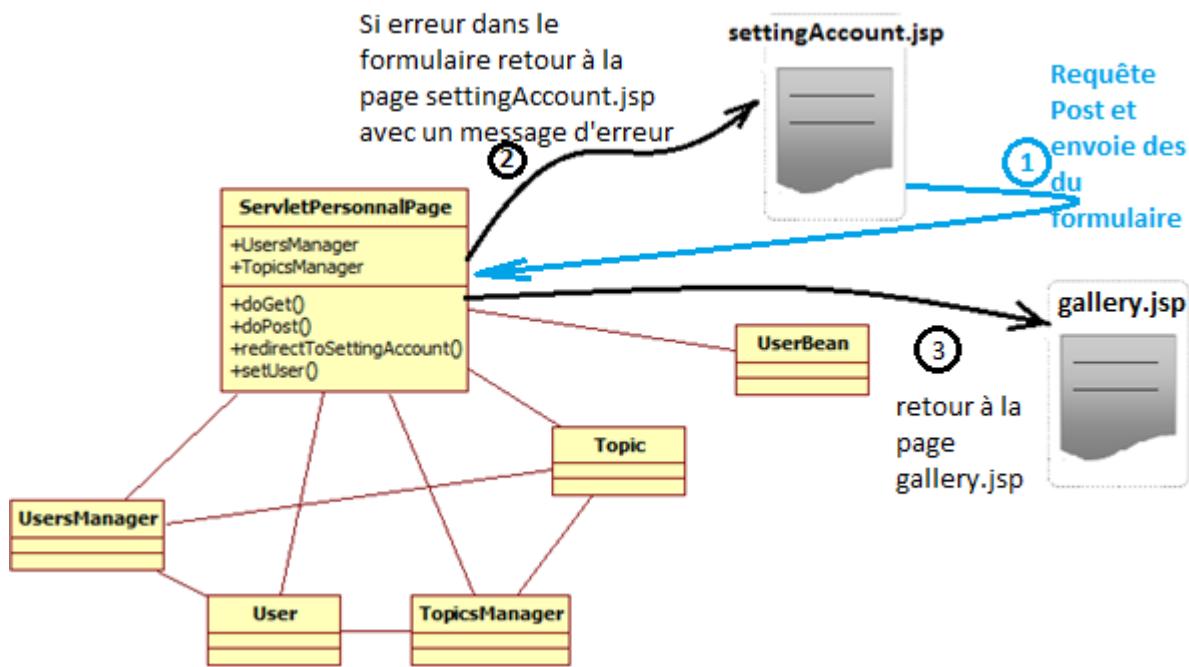


Figure 46 Diagramme de UML de la servlet gérant la modification des données client et interaction avec ses pages correspondantes

### 8.12.2 – page « My Media »

La page « `My Medias` » sert à afficher les médias de l'utilisateur. Lorsque l'utilisateur clique sur l'onglet « `My Medias` », son browser envoie une requête GET à « `UserMediaServlet` ». Cette servlet demander à Instagram les données de l'utilisateur, données qu'elle va parsée pour en extraire un liste d'url des photos « `uploader` » par le dit utilisateur. Cette liste d'url sera insérée dans la session de l'utilisateur et récupérée par la page « `myMedia.jsp` » qui se chargera de les afficher.

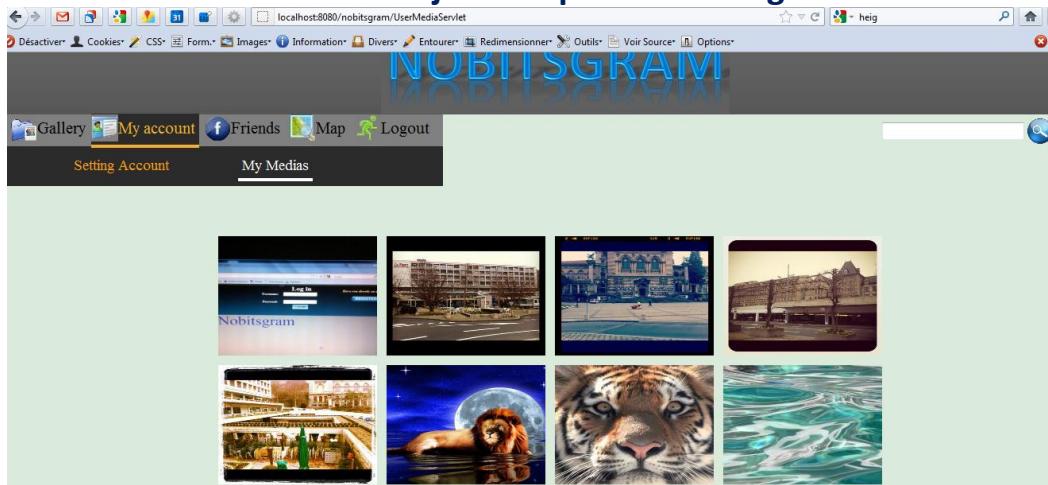


Figure 47 page My Medias

## 8.13 – Onglet Map

### 8.13.1 – Page Photo Map

Cette page affiche une map centrée par défaut sur la position de l'utilisateur. La map utilisée est celle de google. Nous utilisons la version v3 de l'API google map. La map peut être utilisée sans la clé (code alphanumérique récupérer après s'être enregistrer sur google à l'adresse suivante :

<http://code.google.com/intl/fr/apis/maps/signup.html> ). Dans le cas de notre application, nous avons décidé d'utiliser une clé, ce qui permet à google de nous contacter directement lorsqu'il y aura un problème avec l'utilisation de son API. L'affichage de la map nécessite d'écrire du code JavaScript dans la page qui se chargera de l'afficher en l'occurrence « map.jsp ». En suivant le tutorial à l'adresse suivant <http://code.google.com/intl/fr/apis/maps/documentation/javascript/tutorial.html> on se rend compte que l'utilisation n'est pas aussi difficile qu'on pourrait l'imaginer. A partir de ce tutoriel et en faisant quelques autres recherches on arrive facilement à prendre en main google map API. Ainsi, nous pouvons récupérer la position géographie du clic de la souris sur la map et faire un traitement avec cette données. Le traitement effectué est d'afficher les photos d'Instagram se trouvant dans un rayon de 1km de la position cliquée.

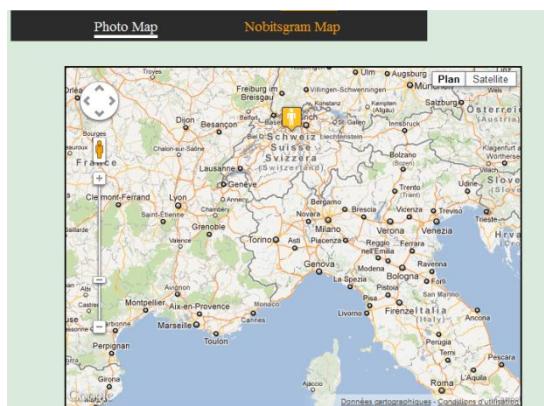


Figure 48 map centrée sur la position de l'utilisateur

## Projet de diplôme Nobitsgram

Lorsque l'utilisateur clique sur une région de la map, une série de photo ayant été prise dans un rayon de 1km est affichée à droite de la map.

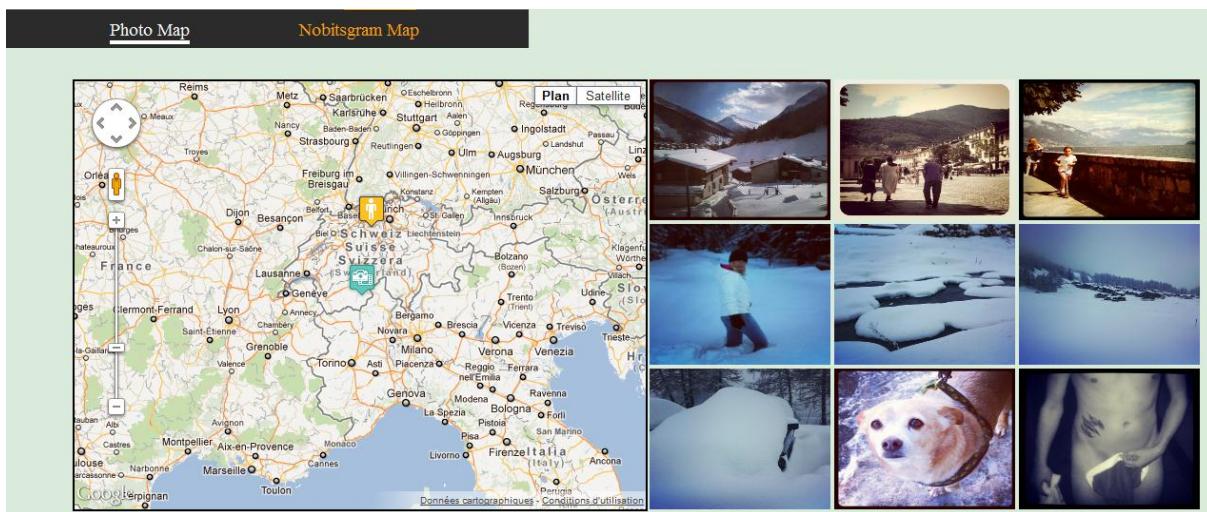


Figure 49 affichage de la map et des photos se trouvant dans un rayon de 1km de l'endroit où se trouve l'icône bleu

Lorsqu'aucune photo n'est trouvée, un message d'erreur est affiché indiquant qu'aucune photo n'a été trouvé dans cette zone de rayon 1km.

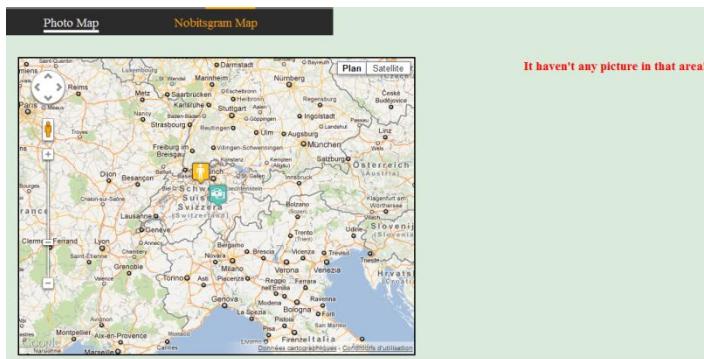


Figure 50 Affichage de l'erreur lorsqu'aucune photo n'a été trouvée dans un rayon de 1km sur l'endroit cliqué

Pour pouvoir afficher les photos, une première chose est de récupérer les coordonnées géographiques de la partie cliquée sur la map. Cette récupération se fait à l'aide de JavaScript. JavaScript possède déjà une méthode permettant de récupérer la position du clic de la sourie. Après la récupération de ces données géographique avec cette méthode, nous forgeons une url dans laquelle nous insérons ces coordonnées comme paramètre et nous envoyons une requête à MapServlet. MapServlet récupère les paramètres se trouvant dans la requête et de son côté forge une url qu'elle utilisera pour contacter le serveur d'Instagram. Cette url aura le format suivant :

```
https://api.instagram.com/v1/media/search?lat=LATITUDE&lng=LONGITUDE
&access_token=ACCESS-TOKEN
```

Cette url demande Instagram les photos se trouvant dans un rayon de 1km de la position donnée par LATITUDE et LONGITUDE. Par défaut, le rayon est de 1km. On peut préciser le rayon en ajoutant le paramètre « distance ». Le rayon maximal traité est de 5km.

MapServlet va parser la réponse reçue d'Instagram pour en extraire la liste d'url des photos. Lorsque la liste est vide, MapServlet forge un message d'erreur qu'elle insère dans la session. La liste va être ensuite insérée dans la session de l'utilisateur afin qu'elle puisse être récupérée par la « map.jsp ». Après l'insertion de la liste dans la session, MapServlet contacte « map.jsp ». « map.jsp » extrait la liste de la session et affiche les images. Lorsque la liste est vide, « map.jsp » teste voir si elle a reçu un message d'erreur venant de la servlet. Si oui, elle affiche le message et aussitôt le supprime de la session.

### 8.13.2 – Page Nobitsgram Map

Cette page se charge d'afficher tous les utilisateurs nobitsgram sur une map. L'utilisateur courant sera repérable par une icône de couleur bleue, les utilisateurs connectés eux auront une icône de couleur verte et tous les autres auront une icône de couleur orange. Un clic sur chaque utilisateur fait apparaître une info bulle dans laquelle se trouve les informations basiques sur l'utilisateur en question.



Figure 51 affichage des utilisateurs de nobitsgram ainsi que leur état de connexion

Pour réaliser ce service, la classe NobitsgramMapServlet qui contrôle cette vue, récupère la liste de tous les utilisateurs de nobitsgram à l'aide de la méthode « findAll() » de la classe UserManager. Cette liste est ensuite envoyée à « nobitsgramMap.jsp ». Dans le code JavaScript qui se charge d'afficher la map, nous créons un marqueur pour chaque utilisateur et son info bulle correspondant. Pour l'affichage des icônes, nous testons si l'utilisateur est connecté ou pas, si il l'est nous assignons une icône verte à l'utilisateur si non, une icône orange.

### 8.14 – Onglet « Friends »



Figure 52 Onglet "Friends" et ses sous onglets

Les services d'interactions avec les utilisateurs de nobitsgram et éventuellement d'autres utilisateurs d'Instagram sont offerts à travers l'onglet « Friends ».

### **8.14.1 - Page « My contacts »**

La page « My contacts » affiche la liste des contacts (followers, followings) qui ont aussi un compte nobitsgram. Pour y parvenir, la servlet « FriendServlet » récupère auprès d'Instagram la liste des followers et followings de l'utilisateur. Elle concatène ces deux listes. Ensuite elle la filtre à partir de la liste de tous les utilisateurs nobitsgram. Pour le filtrage de la liste, elle prend chaque utilisateur de la première liste (followings et followers), et teste s'il se trouve dans la liste des utilisateurs de nobitsgram. S'il s'y trouve, alors on insère cet utilisateur dans une liste vide créée au début. Ensuite cette liste sera envoyé à la page « friendsPage » pour qu'elle affiche ces utilisateurs nobitsgram ainsi que leurs données. Notons que nous avons décidé d'afficher au lieu du username nobitsgram celui d'Instagram.



Figure 53 page affichant la liste des contacts Instagram ayant un compte nobitsgram

### **8.14.2 Page « My topics users »**

Cette page a pour rôle d'afficher la liste des utilisateurs nobitsgram qui ont le même topic que l'utilisateur courant. Pour y parvenir la classe « ListTopicUserServlet » récupère la liste de tous les topics de l'utilisateur et l'envoie à la page « pageListTopicUser.jsp ». Cette page se chargera d'afficher pour chaque topic se trouvant dans la liste récupérée, tous les utilisateurs se trouvant sa liste « users ». En effet, l'entité Topic définit une liste d'utilisateur la partageant comme « topic ».

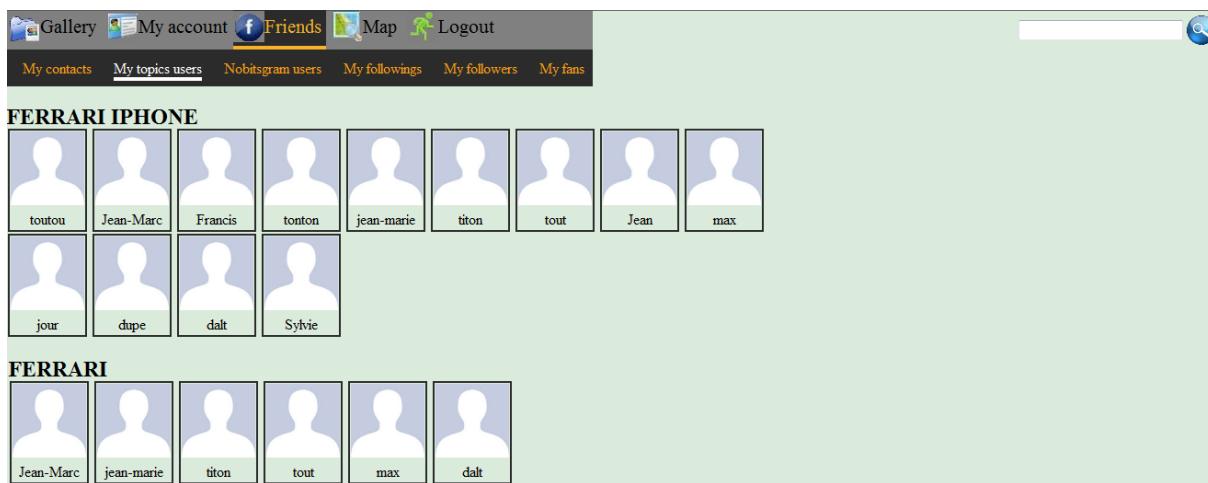


Figure 54 page montrant la liste des personnes partageant le même topic que l'utilisateur courant

### 8.14.3 - Page Nobitsgram users

Cette page affiche la liste de tous les utilisateurs connectés. Pour afficher cette liste, « UsersActualyConnectServlet » (servlet contrôllant la vue de cette page), demande juste au travers d'une requête JPQL effectuée dans la classe « UsersManager » la liste de tous les utilisateurs qui ont leur attribut « isConnected » à « true ». En effet, lorsqu'un utilisateur se connecte à son compte, son attribut « isConnected » est mis à « true » et lorsqu'il se déconnecte celui-ci est mis à « false ». Cette liste sera envoyée à la page « usersConnectedPage.jsp ». Cette dernière récupère la liste et affiche tous les utilisateurs s'y trouvant. Une amélioration qu'on pourrait apporter à cette page est de pouvoir interagir en envoyant un message à une personne donnée de la liste.



Figure 55 affichage de la liste de toutes les personnes connectées sur nobitsgram

### 8.14.4 – Page « My followings »

Cette page a pour rôle d'afficher la liste de toutes les personnes ainsi que leurs photos récentes que l'utilisateur « follow » (suit) sur Instagram. La récupération de cette liste se fait au travers de la servlet « MyFollowingsServlet ». Celle-ci à partir de l'url forgée à l'aide de l'access token et de l'« id » Instagram de l'utilisateur demande au serveur d'Instagram de lui envoyer toutes les personnes que l'utilisateur « follow ». La réponse reçue sera parsée afin d'en extraire une liste d'utilisateur qui sera renvoyée à la page « myFollowings.jsp ». C'est cette page qui se chargera d'afficher toutes les personnes que l'utilisateur suit.

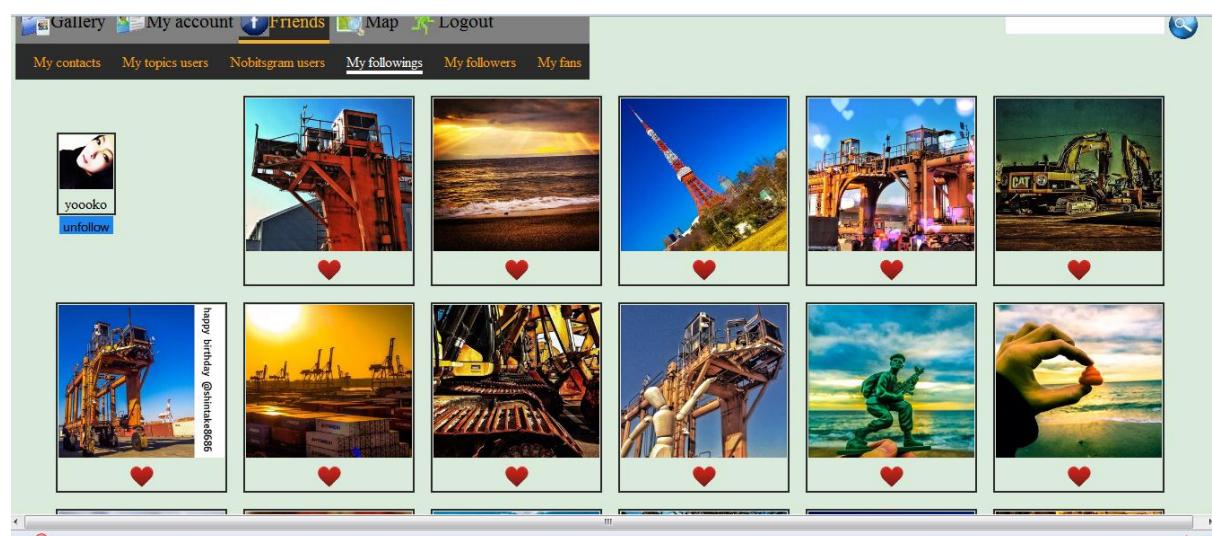


Figure 56: page affichant la liste et les photos de toutes les personnes que l'utilisateur suit

## Projet de diplôme Nobitsgram

Cette page offre la possibilité de ne plus suivre un utilisateur ou soit de « likée » les photos qui nous plaisent. Lorsque l'utilisateur clique sur le bouton « unfollow » pour indiquer à Instagram qu'il ne veut plus suivre la personne concernée, une requête Post en ajax est forgée et envoyé à Instagram. Le format de la requête est le suivant :

```
Avec comme paramètre action=unfollow
https://api.instagram.com/v1/users/ID-
Utilisateur_qu_on_veut_unfollow/relationship?access_token=ACCESS-TOKEN
```

### 8.14.5 – Page « My Followers »

Le but de cette page est d'afficher toutes les personnes qui « follow » l'utilisateur, indépendamment qu'il soit possible que certaines de ces personnes sont « follow » par le dit utilisateur.

« myFollowers.jsp » gère la vue de cette page. Elle est contrôlée par « MyFollowerServlet ». Ainsi, c'est cette dernière qui envoie la liste des « followers » de l'utilisateur à « myFollowers.jsp ». La liste des « followers » est récupérer à partir d'Instagram au travers de la requête l'url :

```
https://api.instagram.com/v1/users/3/followed-by?access_token=ACCESS-TOKEN
```

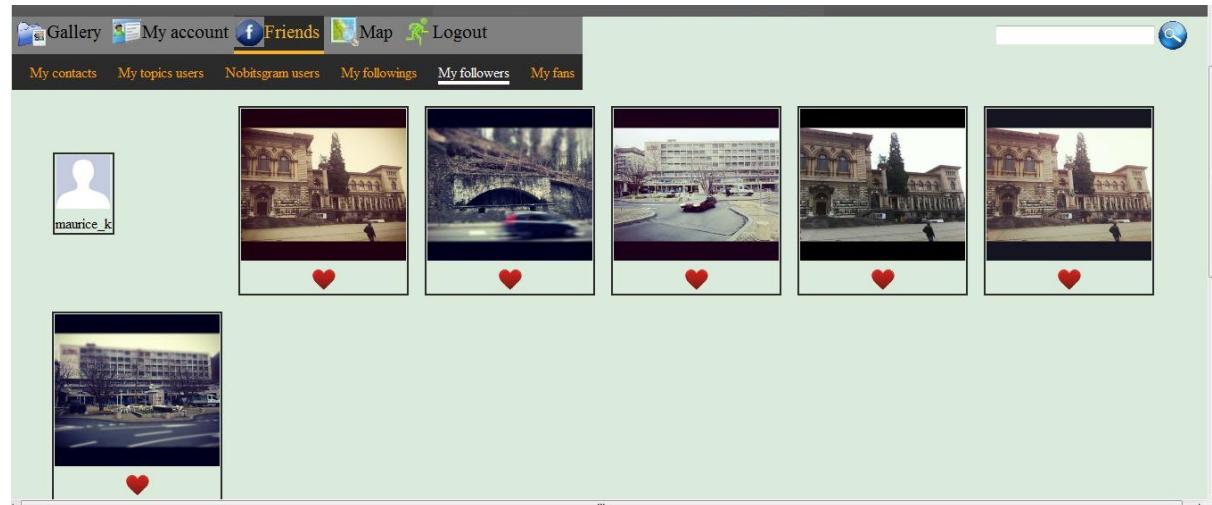


Figure 57 Affichage des followers de l'utilisateur

### **8.14.6 – Page « My Fans »**

Cette page affiche la liste des « followers » de l'utilisateur, mais que ce dernier ne « follows » pas, c'est-à-dire qui ne se trouve pas dans la liste des « followings ». Donc, c'est la liste des personnes qui « suivent » l'utilisateur sans que celui ne les « suivent ». Pour avoir cette liste, « MyFanServlet » (servlet contrôlant cette page) supprime tout simplement de la liste des « followers » les éléments se trouvant dans la liste des « followings ». Ensuite la liste ainsi filtrée est envoyée à « myFans.jsp » pour l'affichage.

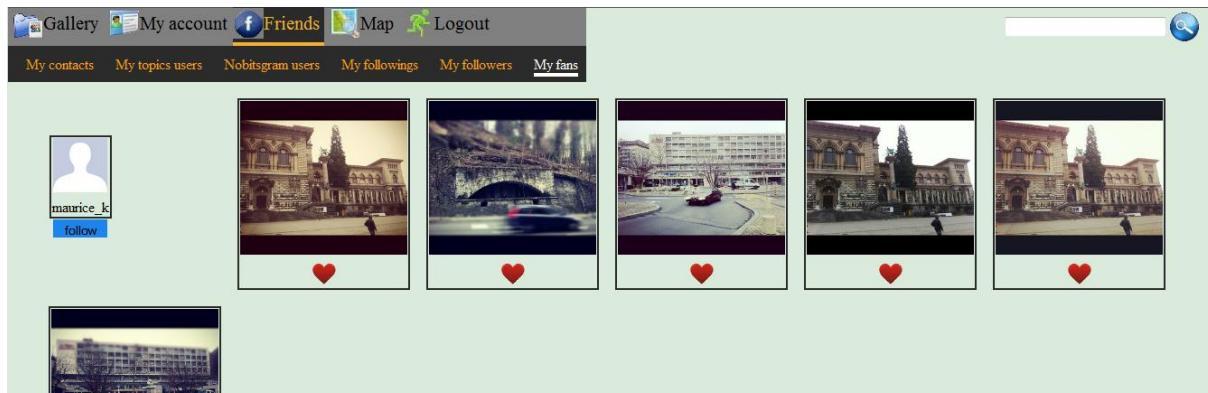


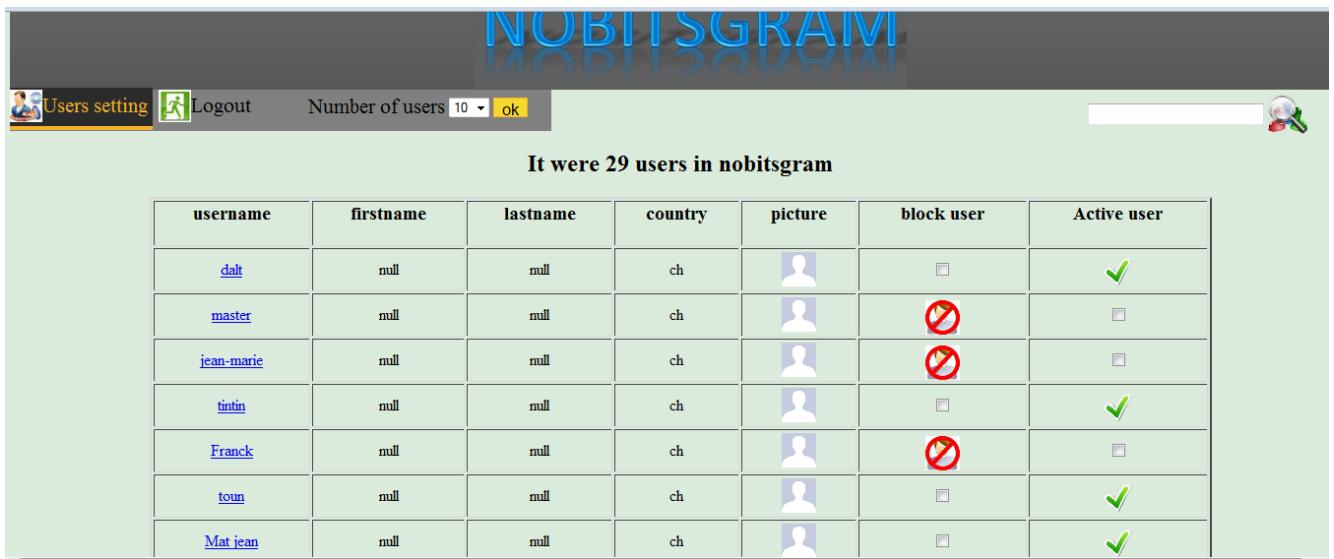
Figure 58: affichage de la liste des "fans" de l'utilisateur

# 9 – Administration de Nobitsgram

L'administration de l'application se fait au travers d'une url spéciale. Seul l'utilisateur qui a le rôle d'administrateur peut y accéder. La page de login de l'administrateur se présente comme suit :



Cette page est contrôlée par la servlet « LoginAdminServlet ». Après que cette dernière eu authentifié l'administrateur, elle le redirige vers une page où l'administrateur pourra avoir une vue d'ensemble sur tous les utilisateurs.



username	firstname	lastname	country	picture	block user	Active user
<a href="#">dalt</a>	null	null	ch		<input type="checkbox"/>	
<a href="#">master</a>	null	null	ch			<input type="checkbox"/>
<a href="#">jean-marie</a>	null	null	ch			<input type="checkbox"/>
<a href="#">tintin</a>	null	null	ch		<input type="checkbox"/>	
<a href="#">Franck</a>	null	null	ch			<input type="checkbox"/>
<a href="#">toun</a>	null	null	ch		<input type="checkbox"/>	
<a href="#">Mat jean</a>	null	null	ch		<input type="checkbox"/>	

**Figure 59: page d'accueil de l'administration de nobitsgram**

L'administrateur peut voir toutes les informations sur un utilisateur donné. Pour cela il suffit qu'il clique sur le « username » de l'utilisateur voulu.

L'administrateur a la possibilité de bloquer un utilisateur ou d'activer un utilisateur. Pour cela il suffit qu'il active le « checkbox » correspondant à la colonne et appui sur le bouton « submit » en bas de la page. Un utilisateur bloqué ne peut être que activé, et un utilisateur activé ne peut être que bloqué.

L'administrateur à la possibilité de naviguer dans les pages au travers des boutons se trouvant en bas de la page.

## Projet de diplôme Nobitsgram

<a href="#">Jean</a>	null	null	ch		<input type="checkbox"/>	
<a href="#">maestro</a>	null	null	ch			<input type="checkbox"/>
<a href="#">jour</a>	null	null	ch		<input checked="" type="checkbox"/>	
<a href="#">dupond</a>	null	null	ch			<input type="checkbox"/>
<a href="#">dupe</a>	null	null	ch			<input checked="" type="checkbox"/>
<a href="#">max</a>	null	null	ch			<input type="checkbox"/>
<a href="#">toto</a>	null	null	ch		<input type="checkbox"/>	
<a href="#">toutou</a>	null	null	ch			<input type="checkbox"/>

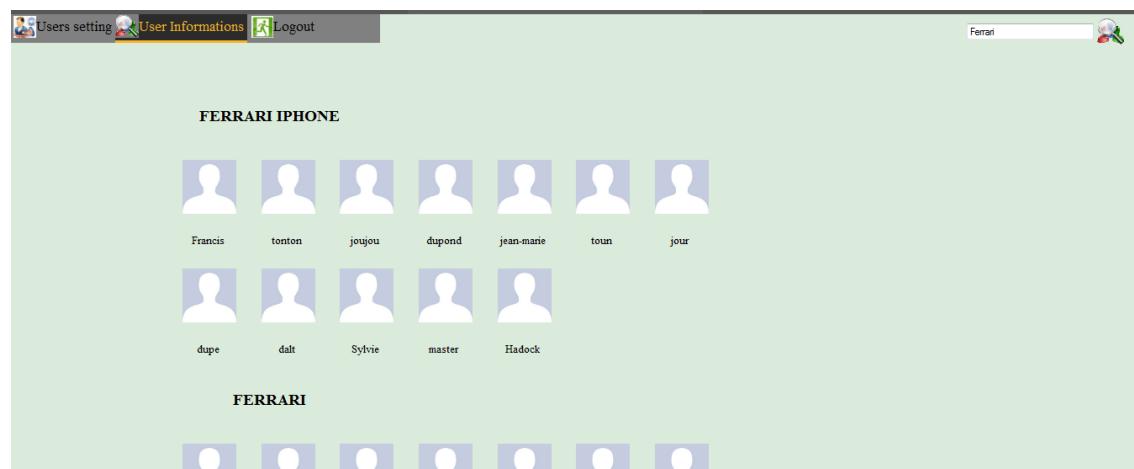
**Submit**

« « NOBITSGRAM » »

1 2 3

Figure 60: Vue du bas de la page d'accueil d'administration

L'administrateur peut aussi saisir un mot clé et voir tous les utilisateurs ayant utilisé ce mot clé dans leurs centres d'intérêts.



The screenshot shows a search interface where the word "FERRARI" has been entered into a search bar. Below the search bar, there are two sections of user profiles. The first section is titled "FERRARI IPHONE" and lists users: Francis, tonton, joujou, dupond, jean-marie, toun, jour. The second section is titled "FERRARI" and lists users: dupe, dalt, Sylvie, master, Haddock. Each user profile includes a small placeholder image and their respective usernames.

Figure 61: Affichage de la liste de utilisateurs ayant utilisé "Ferrari" dans leurs centres d'intérêt



This screenshot displays the detailed information for the user "jean-marie". At the top, it shows the user's profile picture and basic details: firstname (null), lastname (null), Street number (null), Street (null), city (null), Zip code (null), country (ch), Date of count create (Fri Jan 13 14:40:21 CET 2012). Below this, there are four metrics: count of connection in the month (1), count all connection (1), count of search in the month (0), and count all search (0).

Figure 62: Page montrant les données de l'utilisateur "jean-marie"

## 10- Conclusion

---

Ce travail de diplôme a été très passionnant et intéressant, dans le sens où il nous a permis de prendre en main certaines technologie (JavaScript, JSON, Glassfish,CSS...). Ce travail de diplôme est en soit quelque chose de très pratique et qui peut servir plus tard après plusieurs améliorations notamment dans le domaine de la sécurité et de la sa stabilité.

Il est évident que nous avons manqué énormément de temps pour finir la totalité des fonctionnalités que nous avons décidé d'implémenter. Ce manque de temps s'est cruellement fait ressentir dans le rapport, car il y a beaucoup de chose à raconter sur l'implémentation des différentes classes utilisées dans le projet et des difficultés rencontrées. Nous avons passés beaucoup de temps pour l'implémentation des différentes classes, page jsp, code JavaScript, code CSS afin de rendre l'application fonctionnelle. La récupération de l'access token est un point clé du développement de nobitsgram. En effet, sans l'access token, l'application ne pourra pas fonctionner, car elle ne pourra pas dialoguer avec le serveur Instagram.

Grâce à ce travail de bachelor, nous avons appris une nouvelle manière de travailler et une nouvelle manière d'appréhender les problèmes rencontrés. Le seul bémol est le manque de temps pour pouvoir plus exercer durant ce travail de diplôme ces différentes choses apprises. Par contre, ce sont des choses qui nous sont restées ainsi que la manière et les compétences pour réaliser une application web.

Nous remercions le professeur Olivier LIECHTI qui a su être patient quand il le fallait et a su nous dirigé comme il faut tout au long de ce travail de diplôme.

Avec un peu plus de temps, nous aurons pu améliorer considérablement l'ergonomie, la robustesse et la stabilité de l'application.

# 11- Référence bibliographique et internet

---

## Tutoriel netbean webservices:

- <http://www.abricocotier.fr/4224-faire-un-service-web-avec-netbeans-et-glassfish-le-tutoriel>
- <http://www.objis.com/formation-java/tutoriel-web-service-devellopement-netbeans-tomcat.html>
- <http://download.oracle.com/javaee/6/tutorial/doc/bnayk.html>
- <http://wikis.sun.com/display/Jersey/Overview+of+JAX-RS+1.0+Features>
- <http://www.mkyong.com/webservices/jax-rs/restful-java-client-with-jersey-client/>
- <http://download.oracle.com/javaee/6/tutorial/doc/gilik.html>
- <http://jersey.java.net/nonav/documentation/latest/user-guide.html#getting-started>
- <http://www.vogella.de/articles/REST/article.html>

## Java EE 6 et GlassFish 3

### Using JPA Quering

- <http://www.coderanch.com/t/415118/ORM/java/JPA-Query-where-setParameter>
- <http://schuchert.wikispaces.com/JPA+Tutorial+2+-+Working+with+Queries+1>
- [http://openjpa.apache.org/builds/1.0.2/apache-openjpa-1.0.2/docs/manual/jpa\\_overview\\_query.html](http://openjpa.apache.org/builds/1.0.2/apache-openjpa-1.0.2/docs/manual/jpa_overview_query.html)

### Using JSP/SERVLET

- <http://www.java2s.com/Code/Java/JSP/UsingButtons.htm>
- [http://membres.multimania.fr/siteremyjava/code\\_source/java/java\\_jsp.html](http://membres.multimania.fr/siteremyjava/code_source/java/java_jsp.html)
- [http://www.java2s.com/Tutorial/Java/0400\\_Servlet/ServletLogin.htm](http://www.java2s.com/Tutorial/Java/0400_Servlet/ServletLogin.htm)

### Loging authentication with servlet

- <http://www.roseindia.net/jsp/loginbean.shtml>
- <http://met.guc.edu.eg/OnlineTutorials/JSP%20-%20Servlets/Full%20Login%20Example.aspx#>
- <http://www.roseindia.net/jsp/user-registration-form-using-jsp.shtml>

### RestFull definition

- <http://www.figer.com/publications/REST.htm>
- <http://opikanoba.org/tr/fielding/rest/>

## Projet de diplôme Nobitsgram

- [http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)
- <ftp://ftp-developpez.com/mbaron/soa/rest.pdf>

### Google map api

- <http://code.google.com/intl/fr/apis/maps/documentation/javascript/overlays.html#AddingOverlays>
- [http://code.google.com/intl/fr/apis/maps/documentation/javascript/tutorial.html#Loading\\_the\\_Maps\\_API](http://code.google.com/intl/fr/apis/maps/documentation/javascript/tutorial.html#Loading_the_Maps_API)
  - <http://code.google.com/intl/fr/apis/maps/documentation/staticmaps/index.html#Latlons>
  - <http://google-maps-api-version-3.touraineverte.com/fr/MVC-cercle.html>

### Parseur Jackson pour JSON

- [http://www.cowtowncoder.com/blog/archives/2011/08/entry\\_460.html](http://www.cowtowncoder.com/blog/archives/2011/08/entry_460.html)
- <http://wiki.fasterxml.com/JacksonInFiveMinutes>
- <http://stackoverflow.com/questions/2255220/how-to-parse-a-json-and-turn-its-values-into-an-array>
  - <http://wiki.fasterxml.com/JacksonTreeModel>
  - <http://jackson.codehaus.org/1.0.1/javadoc/org/codehaus/jackson/JsonNode.html>

### Pagination JSP

- <http://www.easywayserver.com/blog/jsp-pagination/>