Spécifications techniques

Client	**Cadre universitaire**	**Projet**	***Gask***
Créé le	**08/02/2015**		
Auteur	**Zaini nabil Et Benhammou amin**	**Version**	1
Destinataires	**Professeur de JEE / Public**		

Table des Matières

1. Contexte

L'objectif de notre projet est de mettre en place une application web des questions/réponses destiné aux informaticiens. Le périmètre de ce projet s'inscrit dans un projet universitaire, cependant dans les envies personnelles on veut l'étendre dans une échelle un petit peu évolué.

2. Fonctions de service

2.1. Fonctionnalité 1

La fonctionnalité de notre application accomplit des données en entrés des différents utilisateurs soit des questions, réponses à ces questions des commentaires ect. Et comme sortie l'utilisateur qui pose une question peut reçoit une notification s'i sa question est répondu ou commenter, afin qu'il donne son avis sur les différents réponses.

2.2. Ajouter une Question (exemple)

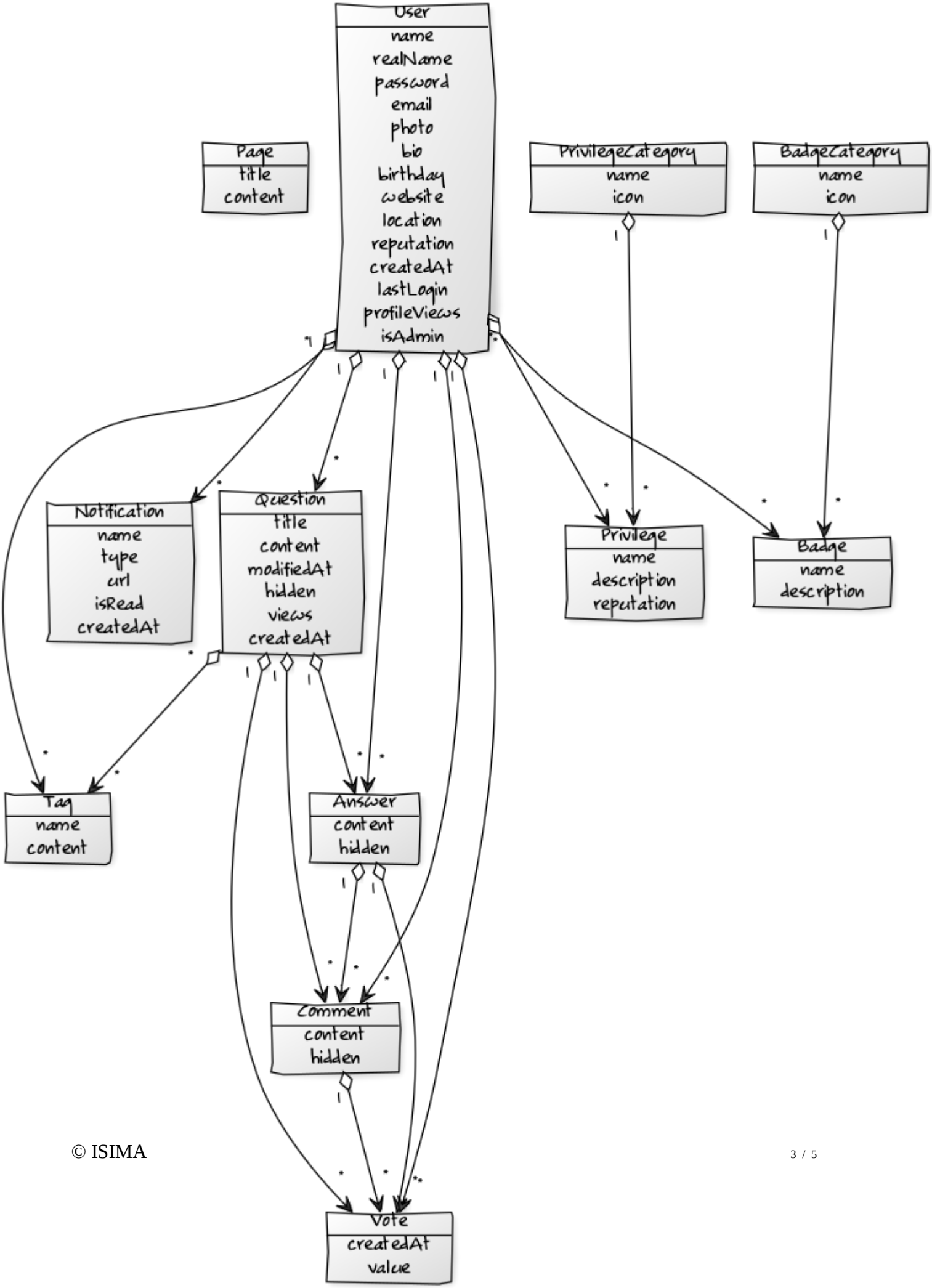
L'utilisateur saisit son contenu via un formulaire de saisie. Quand il clique sur le bouton Ask, le contenu est ajouté dans la base de données après tout un tas de validation. Le formulaire permet la saisie des informations suivantes :

- le contenu ;
- les tags ;

3. Description des données

3.1. Diagramme de classe de notre application

Le diagramme de classes dans notre application va représenter le schéma utilisé en génie logiciel pour présenter les classes du système ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.
Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets.
Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique dans notre cas on a mis juste un diagramme de classe non détaillé. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.
Elles sont finalement instanciées pour créer des objets (une classe est un domaine à objet : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).


User
name
realName
password
email
photo
bio
birthday
website
location
reputation
createdAt
lastLogin
profileViews
isAdmin
Page
title
content
PrivilegeCategory
name
icon
BadgeCategory
name
icon
Notification
name
type
url
isRead
createdAt
Question
title
content
modifiedAt
hidden
views
createdAt
Privilege
name
description
reputation
Badge
name
description
Tag
name
content
Answer
content
hidden
Comment
content
hidden
Vote
createdAt
value

4. Architecture technique

4.1. Architecture applicative

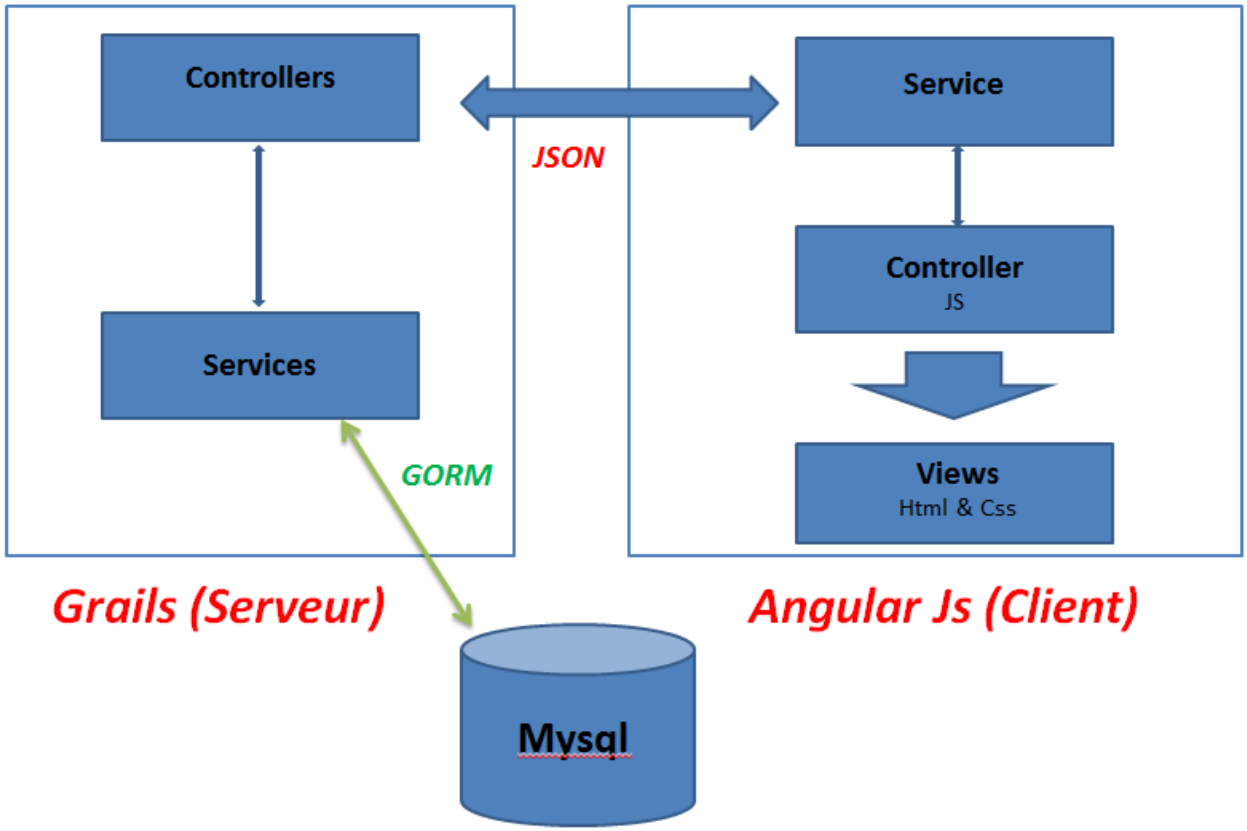
Notre application est découpée en 3 couches :

La couche présentation qui interagit avec l'utilisateur de notre application (Front) est développée en Angular Js. L'intérêt d'utiliser coté client ce Framework de javascript c'est que :

Il est fondé sur l'extension du langage HTML par de nouveaux tags et attributs pour aboutir à une définition déclarative des pages nos web, par opposition à l'utilisation systématique de l'élément 'div' et à la définition les éléments de présentation en javascript. Le code HTML étendu représente alors la partie vue du patron de conception logicielle MVC, auquel Angularjs adhère, avec des modèles appelé «scopes» et des contrôleurs permettant de définir des actions en code javascript impératif. Angularjs utilise une boucle de dirty-checking pour réaliser un data-binding bidirectionnel permettant la synchronisation automatique des modèles et des vues.

La couche métier ou bien notre service web qui est développé en langage groovy coté serveur, l'avantage d'utiliser cette séparation est très importante pour garder les services. Ce web service fournit au client un menu des fonctionnalités. L'échange des informations entre le client et le serveur se fait à travers le format JSON.

La partie couche d'accès aux données est préconfigurés coté serveur dans le fichier Config.groovy. L'interaction des données envoyés par le client vers base de données passe par notre web service fait le traitement et stocke ou récupère des données depuis/vers la base de données de type Mysql. Le fameux GORM et le mapping objet relationnel du framework Grails qui s'appuie sur Hibernate. Celui le responsable de mapper les domaines définit au niveau de service web et les tables relationnelles coté base de données.



5. Règles de codage et conventions de nommage

Les règles de codage qu'on a mis sont un ensemble de règles à suivre pour uniformiser les pratiques de développement de notre application, diffuser les bonnes pratiques de développement et éviter les erreurs de développement "classiques".

Les règles de codage s'articulent autour de plusieurs thèmes, les plus courants étant:

Le nommage et l'organisation des fichiers du code source : on a organisé notre application en deux parties partie back représente le traitement métier et la partie front qui représente l'aspect présentation. En plus notre code est indenté, et les conventions de nommage des variables et les méthodes et très importantes afin d'éviter l'incohérence lors de l'appel d'une fonctionnalité depuis le client. Ainsi que, notre code source gère bien les différentes erreurs.