# Projet HelpDesk

Vous travaillez au sein d'un établissement de formation en tant que Responsable du support aux utilisateurs (formateurs, administratifs). Dans le cadre de votre travail, vous gérez aussi bien les incidents que les demandes d'assistance technique ou fonctionnelle sollicitées par les utilisateurs.

Les demandes d'assistance ou remontées d'incidents ne sont pour l'instant pas informatisées, et les utilisateurs doivent vous contacter directement, par mail ou par téléphone, pour vous communiquer ces informations.

Ce procédé est coûteux en temps, insatisfaisant pour les utilisateurs dont les demandes sont parfois oubliées. Il ne permet pas d'obtenir une traçabilité des actions d'exploitation menées.

Vous avez un temps envisagé la mise en place de solutions existantes (GLPI + OCS), mais ces solutions sont trop complètes, parfois trop complexes pour le SI de votre entreprise.

Vous avez donc décidé de réaliser une application Web permettant de gérer les demandes d'assistance, et dont les fonctionnalités sont adaptées à vos besoins.

Une version initiale a déjà commencé à être implémentée : elle utilise un micro-framework pour la mise en place de MVC, et Bootstrap pour la partie présentation.

## Résumé

| | |
|---|---|
| **Projet initial à utiliser** | • Projet Helpdesk Aada v1.0 sur gitHub<br>• La base de données est dans le dossier app/database du projet |
| **Outils** | • Bootstrap<br>• Micro-framework |
| **Principales fonctionnalités** | • Report d'incidents<br>• Base de connaissances |
| **Livraison 1** | • Fin septembre, par gitHub |

## Détail des fonctionnalités à mettre en place

Les différentes pages devront gérer les droits, et n'être autorisées que pour les acteurs mentionnés.
Tenir compte du fait que l'administrateur a le droit d'accéder à tout ce qui est autorisé à l'utilisateur.

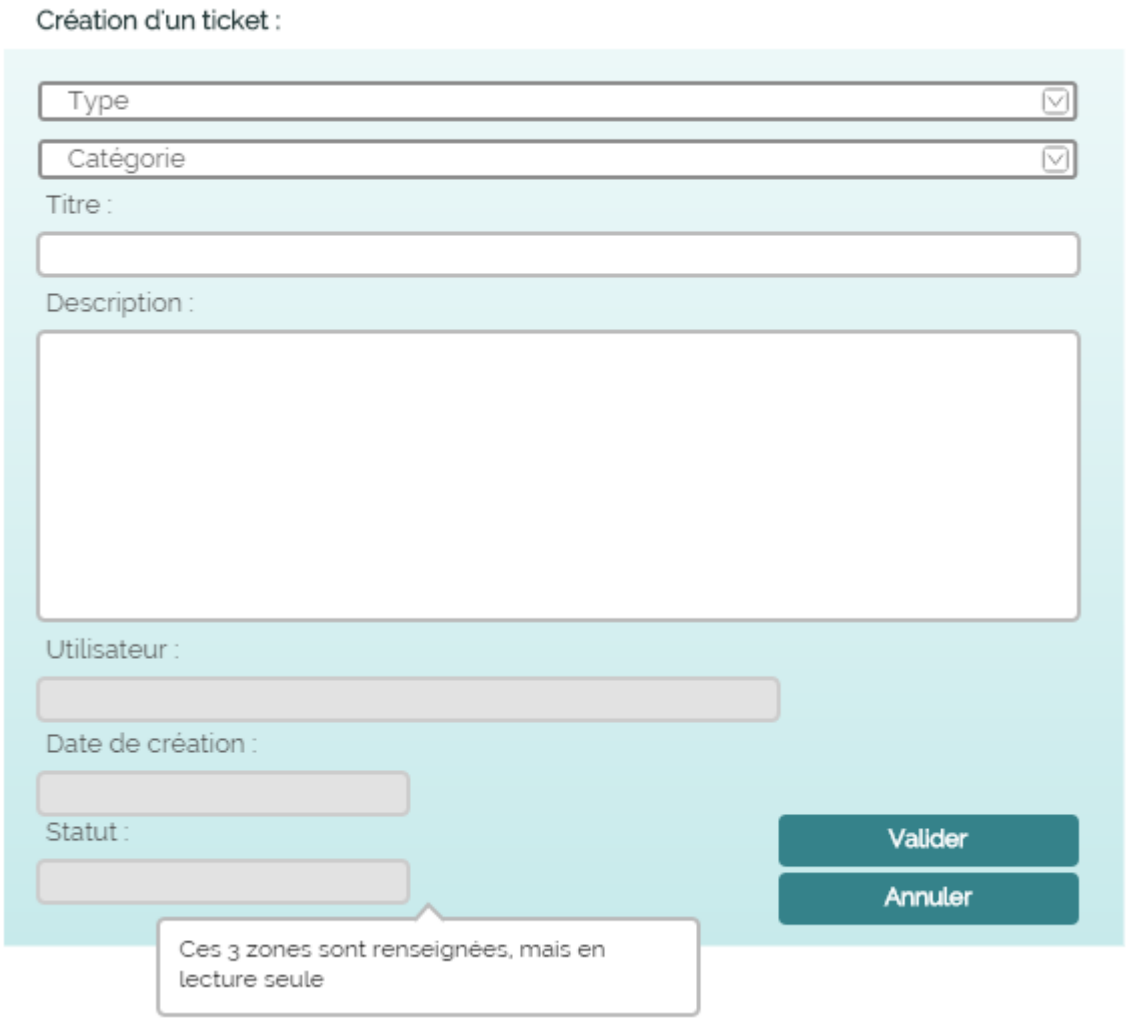
2 des modules de l'application sont à mettre en place :

### -- Module report d'incidents

#### -- Création d'un ticket

- Acteur : utilisateur
- Données : ticket, statut, catégorie, utilisateur

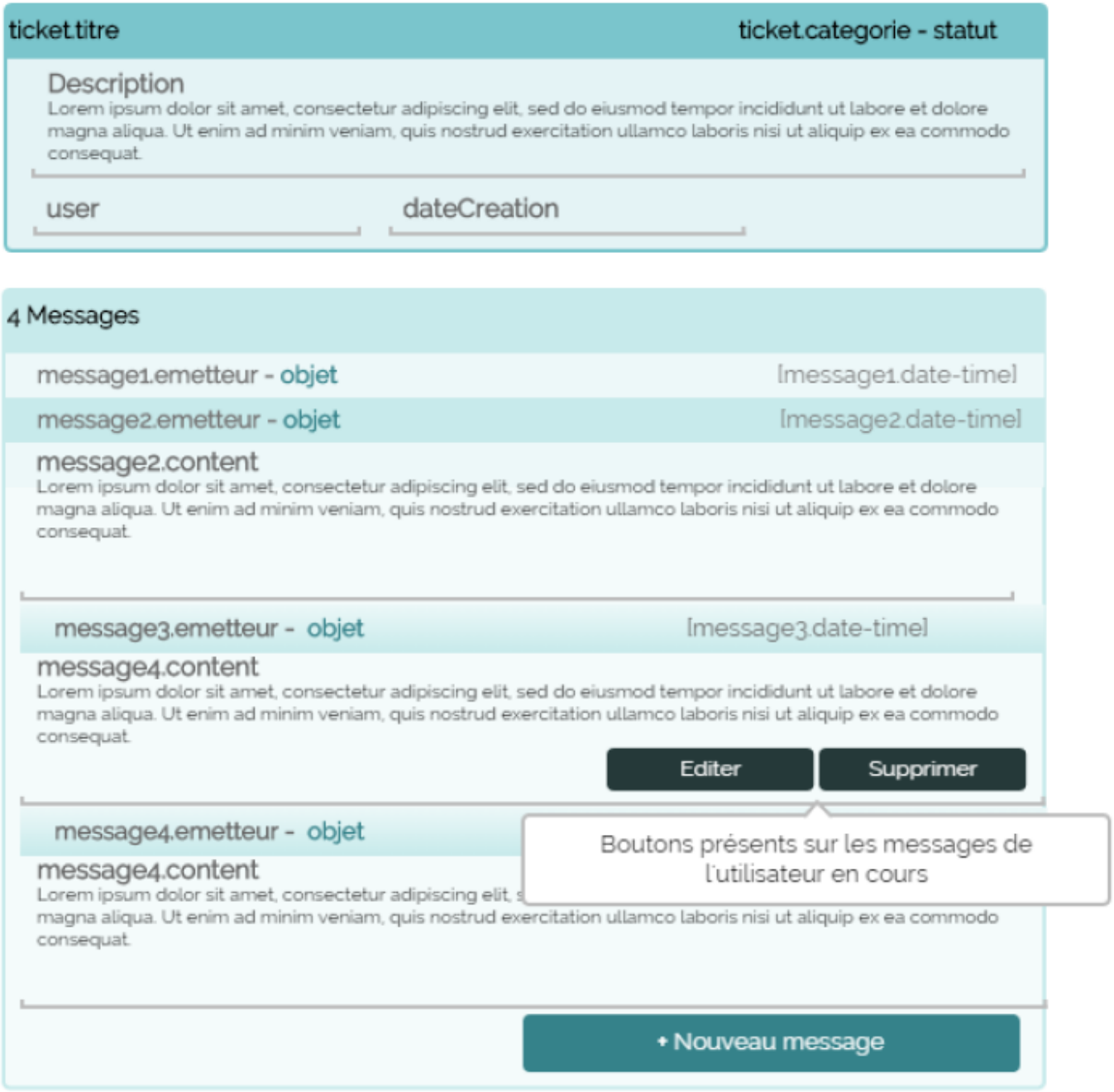
Un utilisateur peut créer un ticket (report d'incident ou demande) dans une catégorie.




**-- Echanges sur ticket**

- Acteurs : Admin et/ou utilisateur
- Données : message, ticket, utilisateur

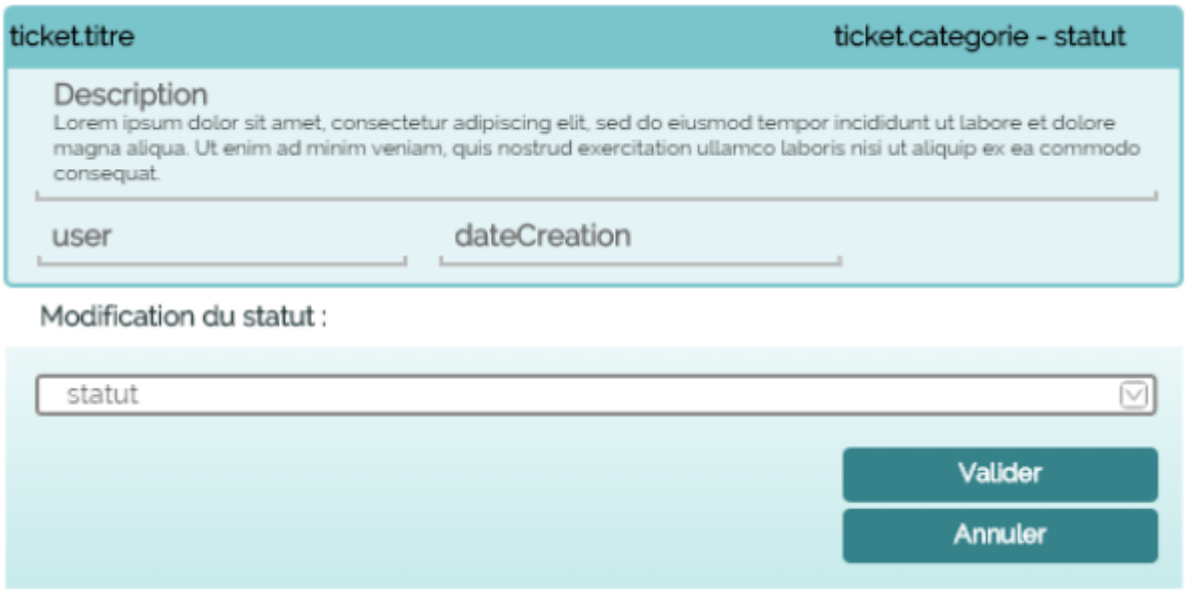
Des messages sont échangés relativement à un ticket, par les utilisateurs. Les messages sont décrits par un objet, un contenu, une date




**-- Modification du statut**

- Acteurs : Admin
- Données : ticket, statut

Les utilisateurs de type Admin peuvent modifier le statut d'un ticket existant




## -- Module Base de connaissances

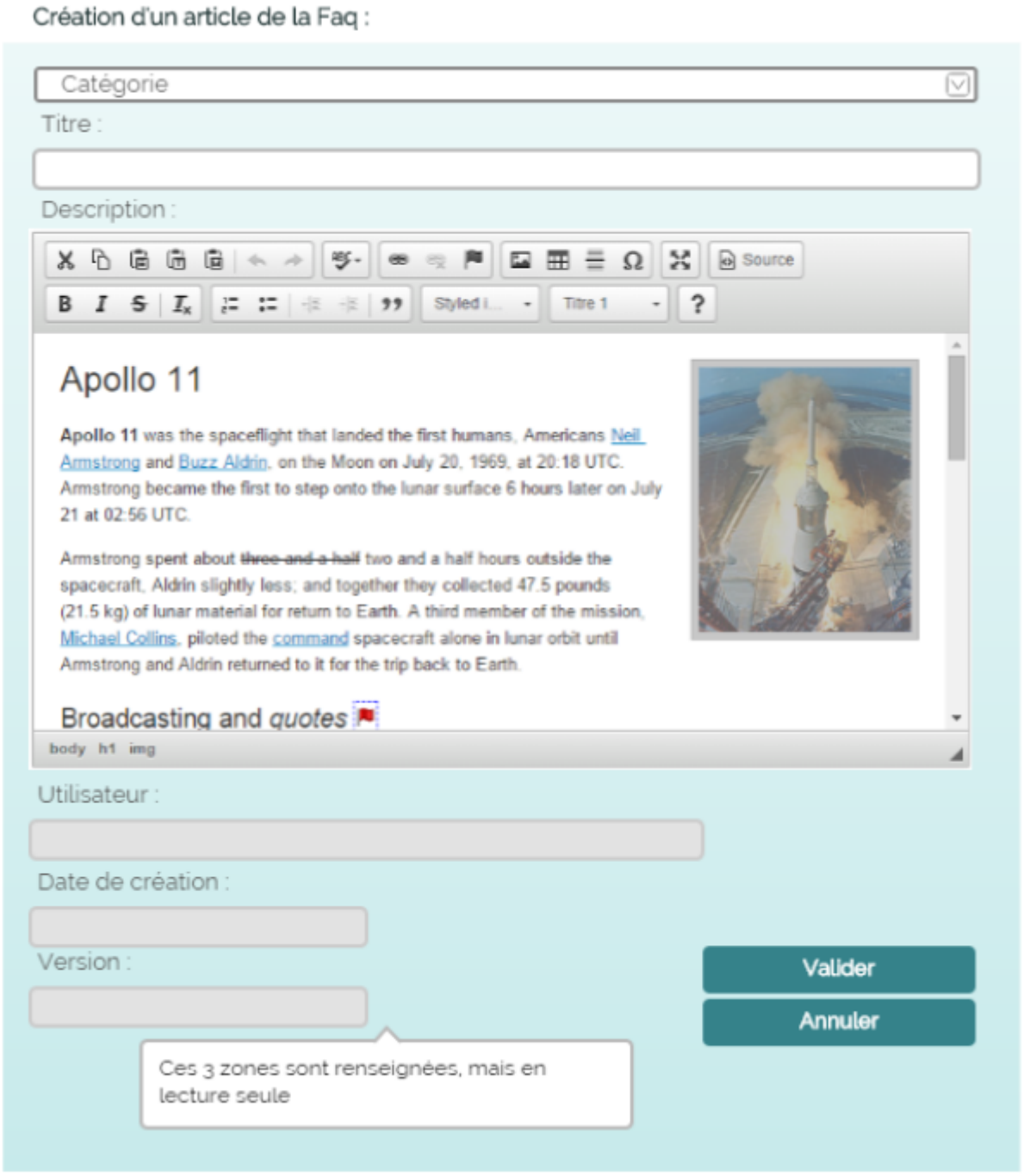
On appelle **article** un enregistrement de la table **FAQ**

### -- Création d'un article

- Acteur : Admin
- Données : faq, catégorie, utilisateur

La base de connaissance est constituée par les utilisateurs de type Admin : il s'agit d'une FAQ.

On utilisera le plugin ckEditor pour faciliter la rédaction du contenu des articles

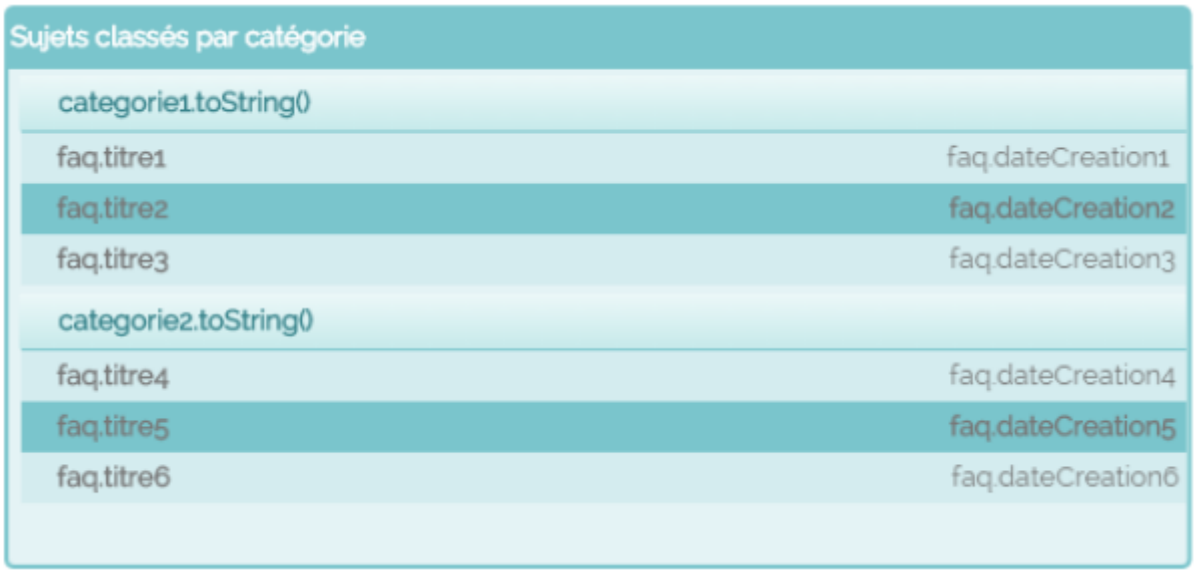



**-- Consultation de la base**

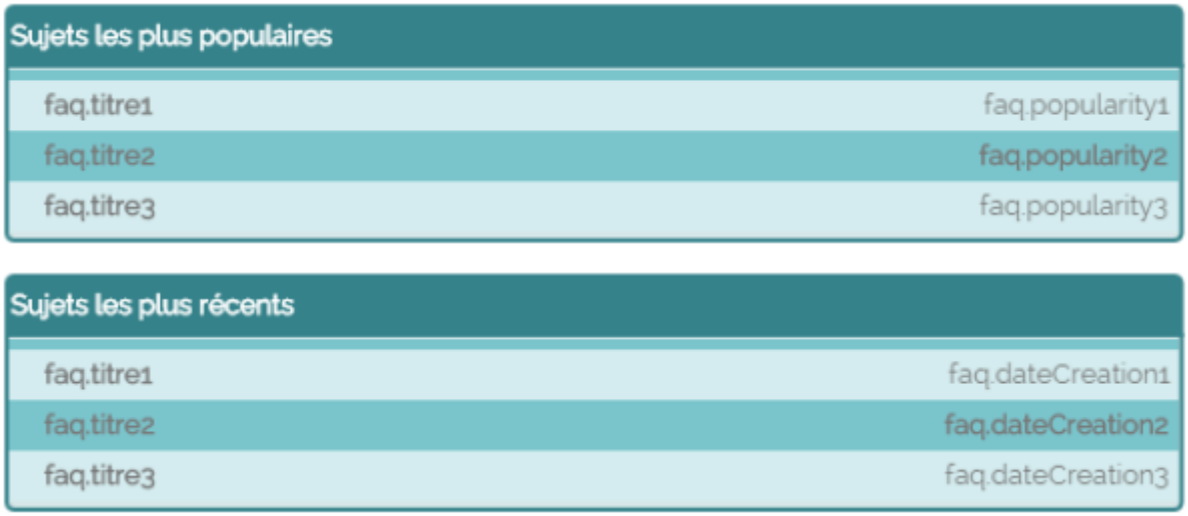
- Acteur : utilisateur
- Données : faq, catégorie

**//TODO** Chaque article listé doit faire le lien (href) vers l'affichage d'un article exposé dans le point suivant.

La base de connaissances est consultable par les utilisateurs, elle est par défaut présentée par catégories :




On mettra également en évidence les articles de la Faq classés par popularité (les 10 plus populaires), et par date de création (les 10 les plus récents) :




**-- Affichage d'un article**

- Acteur : Utilisateur
- Données : faq, catégorie, utilisateur

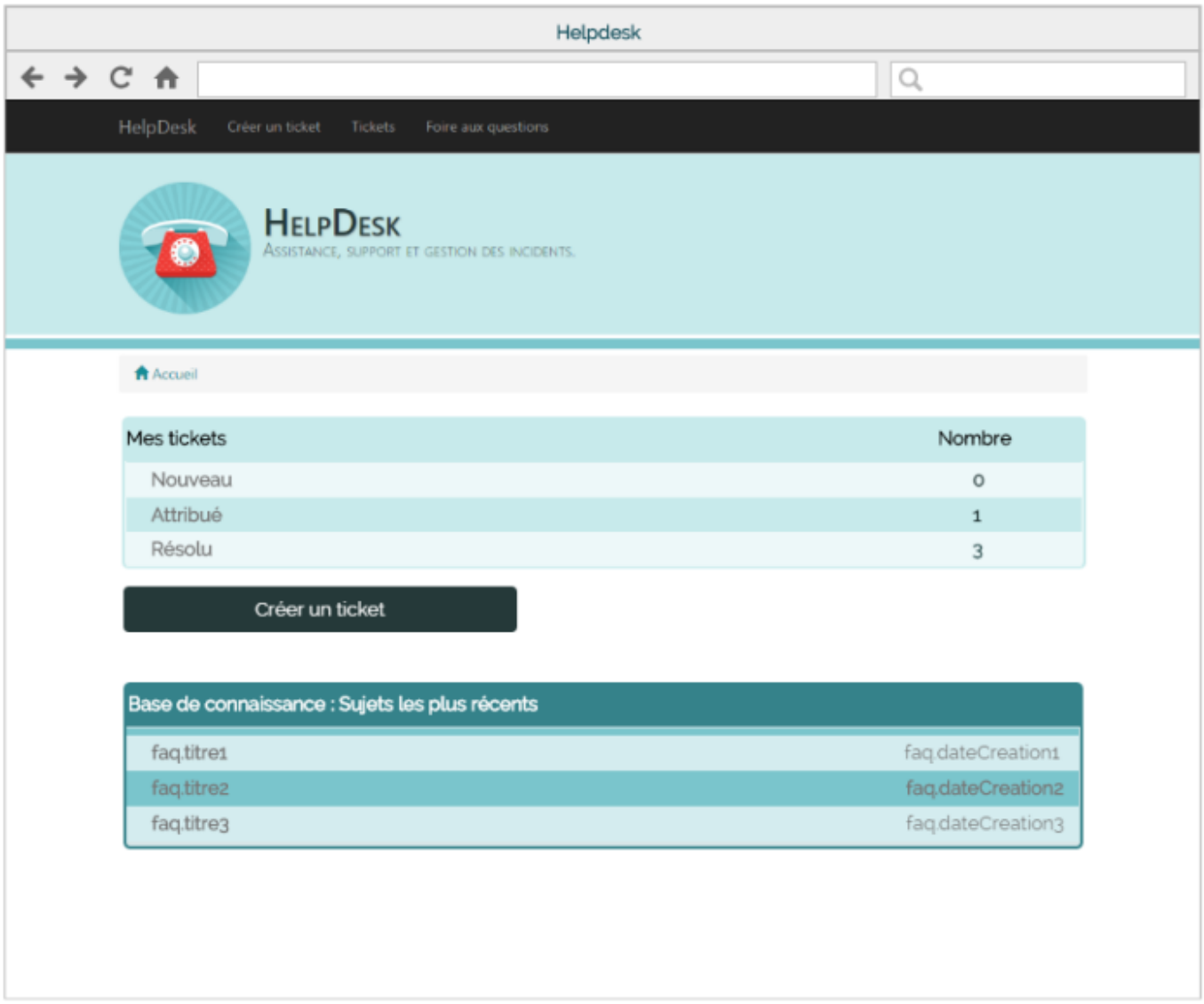
Chaque article de la Faq peut-être affiché par les utilisateur. A chaque affichage, la polpularité (popularity) de l'article est augmentée de 1.




## -- Module Utilisateurs

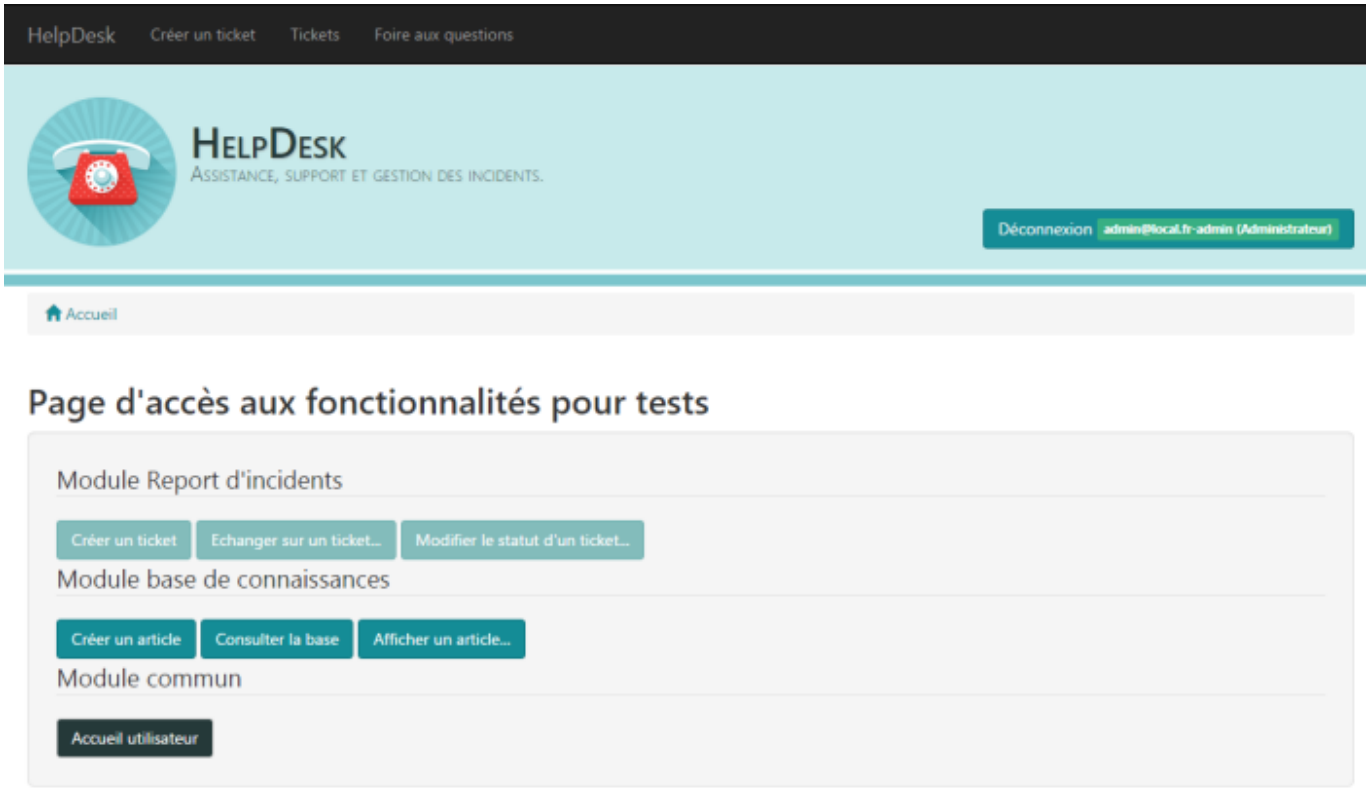
- Acteurs : utilisateur
- Données : ticket, statut, faq

### -- Ecran d'accueil des utilisateurs

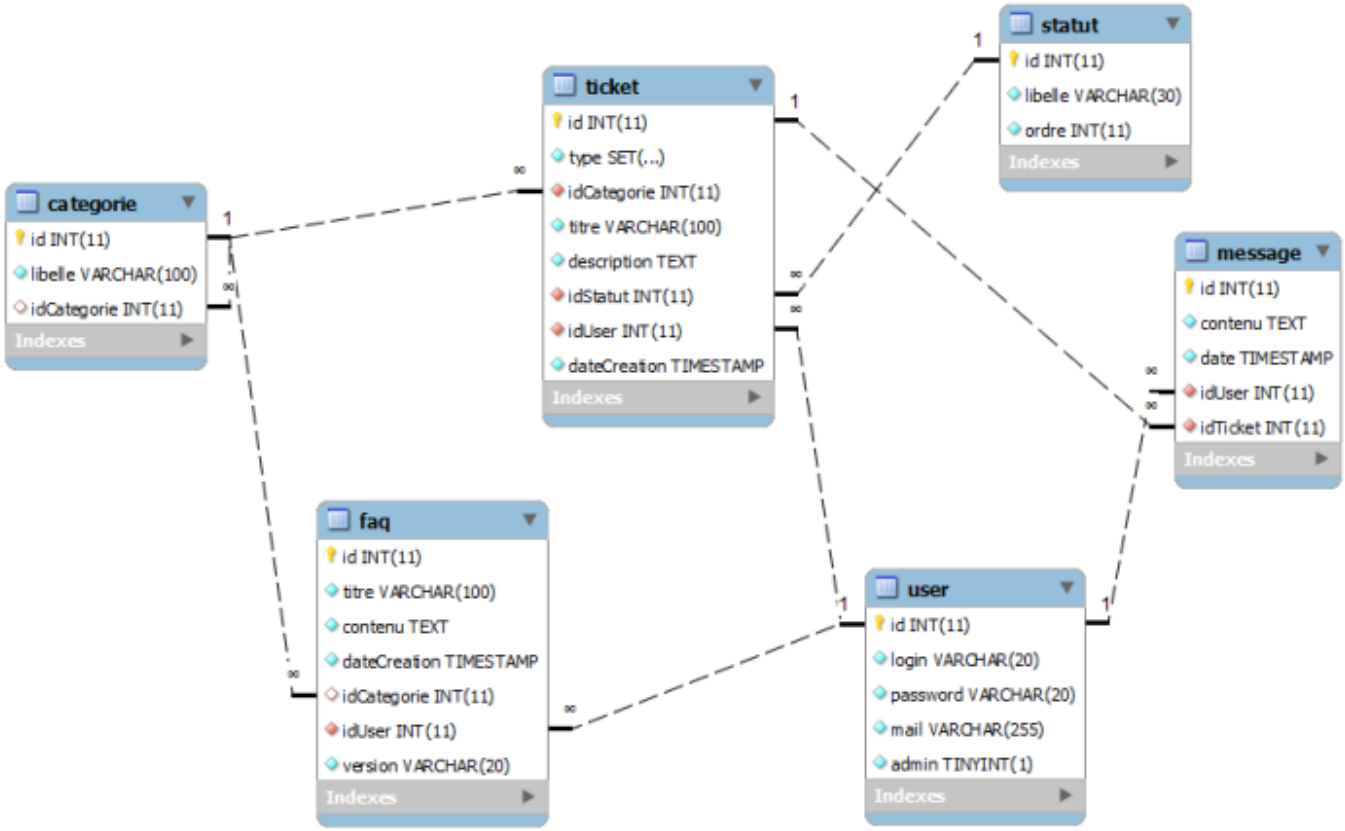



**-- Page de test**

Pour accéder aux fonctionnalités implémentées, créer une page de test :




## Base de données



# Contraintes techniques

- L'application sera développée en PHP objet, et utilisera un micro-framework facilitant les échanges avec la base de données.
- Elle respectera au mieux la séparation des couches (objets Métiers), classes techniques et vues (interfaces web de saisie et d'affichage).
- Elle utilisera la base de données Mysql fournie en annexe. Cette base pourra évoluer en fonction des besoins du développement.
- L'utilisation de scripts côté client (javascript et ajax) devra compléter les validations côté serveur (JQuery en php).
- Bootstrap sera utilisé pour la partie présentation.

# Fichiers

- Projet Github initial à cloner

# Modalités de conduite du travail

- Travail en groupes (2 à 3 personnes)
- Utilisation de gitHub en continu
- Mise en place de tests

# Compléments

## Bonnes pratiques

- Respecter MVC, y compris pour la partie client (javascript)
- Alimenter correctement la base de données en ajoutant des enregistrements valides et en nombre suffisant, mettant en valeur votre travail
- Respecter la Normalisation HTML 5/Css 3
- Structurer les fichiers et dossiers de manière cohérente
- Nommer en respectant les normes et de manière significative (pages, fonctions, variables...)

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
**http://slamwiki2.kobject.net/slam4/helpdesk?rev=1442154764**

Last update: **2019/08/31 14:39**