HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Projet WEBRAILS

Julien Amacher, Romain Maillard

2016

News

Table des matières

1 Introduction

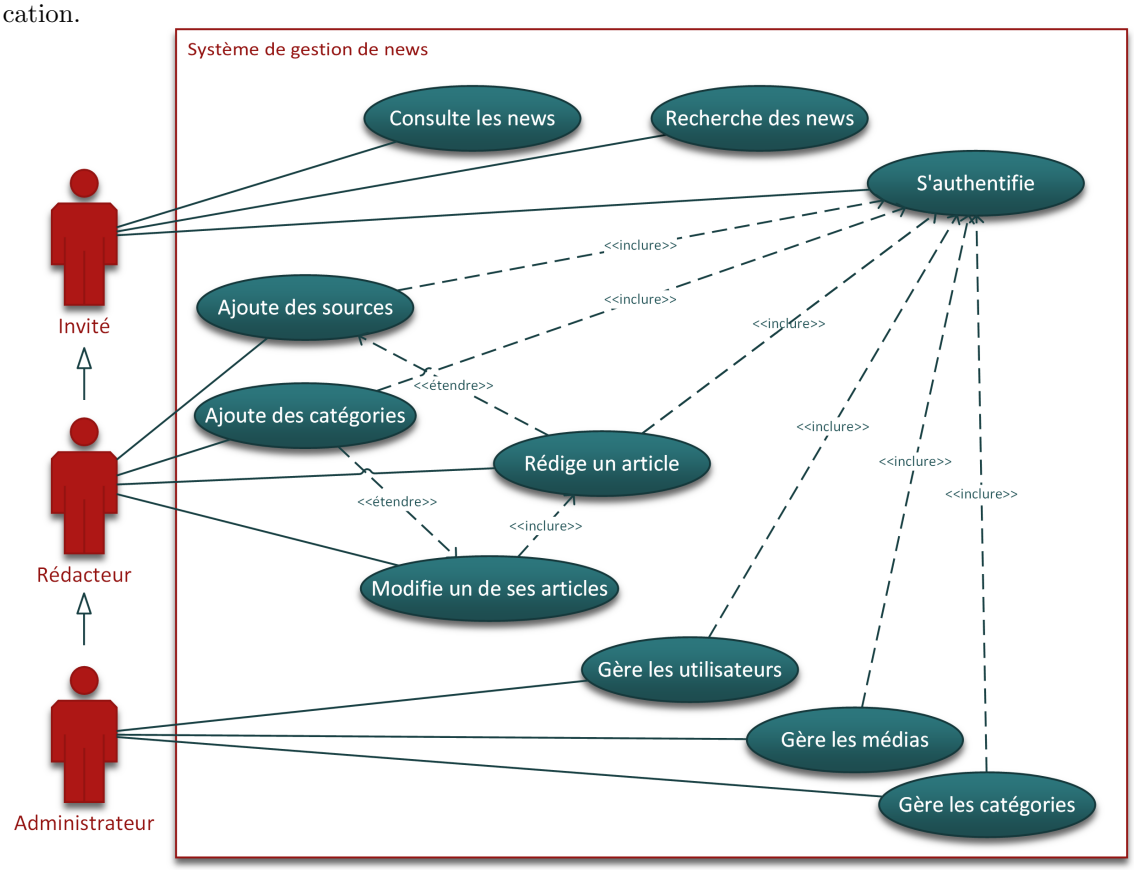
Ce projet a pour but de réaliser une plate-forme permettant d'une part la rédaction de news et d'autre part leur consultation. Les rédacteurs écrivent et modifient des news, en y liant des catégories (par ex : politique) et des sources (par ex : Paul Dupont). Chaque rédacteur travaille pour un média (par ex : France Info)

2 Contraintes

Les contraintes du projet sont les suivantes
- Le langage de programmation est Ruby on Rails (RoR)
- La base de données est MySQL

3 Cas d'utilisation

Ce chapitre décrit les attentes et les fonctionnalités fournies pour chaque type de client de l'application.



3.1 Les invités

3.1.1 Consultent les news

Ils peuvent librement consulter les news qu'ils désirent. Celles-ci sont présentes sur la page principale du site.

3.1.2 S'authentifient

Les invités s'authentifient en fournissant les informations suivantes :
- Leur adresse email
- Leur mot de passe

Leur compte est ensuite identifié et leur rôle leur est attribué (rédacteur ou administrateur)

3.1.3 Recherche des news

Il est possible de rechercher une news par son titre, sa catégorie ou sa source.

3.2 Les rédacteurs

3.2.1 Ajoutent une source à leur news

Une news ayant au minimum une source, le rédacteur de la news peut en créer et en lier. Une news est consistuée d'un titre, d'un chapeau ainsi que d'un contenu.

3.2.2 Ajoutent une catégorie à leur news

Une catégorie pouvant contenir plusieurs news (par exemple : politique), un rédacteur peut spécifier les catégories concernées par sa news.

3.2.3 Rédigent une news

Le rédacteur peut écrire une news. Une news comprend un titre, un chapeau (extrait en gras), un contenu ainsi qu'une liste de sources et est contenue dans au moins une catégorie.
Idéalement, les sources devraient pouvoir être créées 'sur le vif' lors de la rédaction d'une news.

3.2.4 Modifient une news

Le rédacteur peut modifier une de ses news, cela inclut :
- Son titre
- Son chapeau
- Son contenu
- Ses sources liées (en ajouter, en supprimer)
- Ses catégories liées (en ajouter, en supprimer)

3.3 Un administrateur

3.3.1 Gère les utilisateurs

Un administrateur crée, modifie et supprime les comptes des utilisateurs du système.

3.3.2 Gère les catégories

Un administrateur gère les catégories : il peut en ajouter, en modifier et en supprimer.

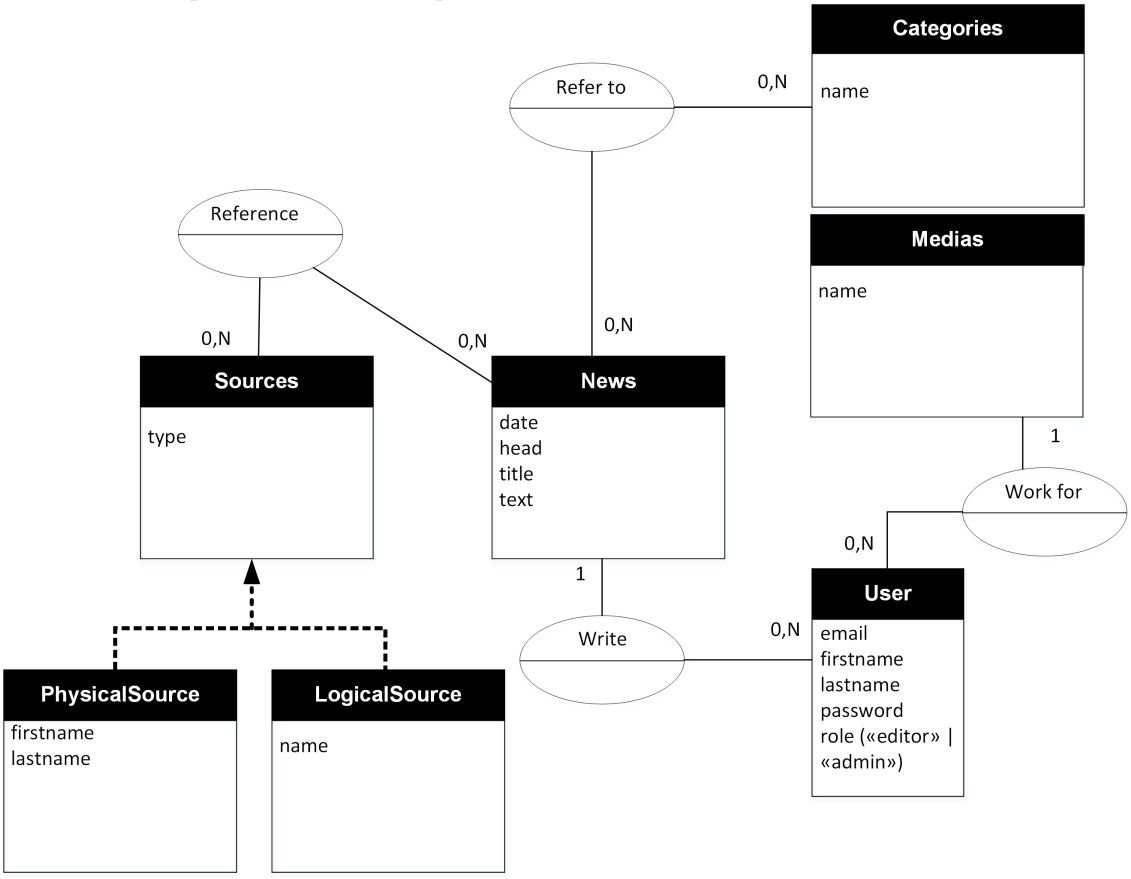
3.3.3 Gère les médias

Un administrateur gère les médias : il peut en ajouter, en modifier et en supprimer. Chaque rédacteur est lié à un média et l'administrateur est le seul à pouvoir modifier le média au quel le rédacteur est rattaché.

4 Base de données

4.0.1 MCD

Le modèle conceptuel des données se présente comme suit :



Contraintes :

- Une news a au minimum une source liée
- Une source fait référence à au minimum une catégorie (politique, économie, etc.)
- Une source peut exister même si aucune news n'y fait référence, idem pour les catégories de news
- Un rédacteur est lié à un média

4.0.2 MPD

Users

- firstname
- lastname

Cancancan gère le rôle de l'utilisateur et Devise gère l'authentification (et y ajoute l'attribut email).

Report

- head
- title
- text

Chaque rapport a une date de création. Nous utilisons le champ created at de Rails.

Sources

- firstname
- lastname
- name
- type

Categories

- name

Medias

— name

5 Itérations

— 3 mai
  — Création de la base de données et des pages squelettes.
— 17 mai
  — Ajout, modification et suppression de news, utilisateurs, catégories et médias fonctionnent.
— 7 juin
  — Fonctionnalité de recherche terminée + droits d'accès implémentés.

6 Fonctionnalités Ajax

La recherche des news se fait via Ajax. La recherche se fait par news, catégorie et source. Le contenu, titre et chapeau de la news sont considérés lors de la recherche.

7 Manuel utilisateur

Disponible en annexe.

8 Configuration CanCanCan

La restriction de l'exécution en fonction des rôles est implémentée comme suit dans le fichier ability.rb :

```
class Ability
  include CanCan::Ability

  def initialize(user)
    user ||= User.new

  if user.role == "admin"
    can :manage, :all
    can :reset_password, User
  end

  if user.role == "author"
    can :manage, Report
    can :manage, Source
  end

  can :read, Report
  can :read, Category
  can :read, Source
  end
end
```

9 Implémentation Ajax

Notre fonctionnalité de recherche a été implémentée en utilisant les mécanismes de requête HTTP accessibles depuis Javascript (Ajax)
Le but est le suivant : lorsqu'un terme de recherche est entré, une recherche est effectuée dans le but de trouver les news, catégories, sources et médias correspondants. A cet effet, la recherche s'opère :

— Le titre des news
— Le chapeau des news
— Le contenu des news
— Le nom des catégories

— Le nom des sources (s'il s'agit d'une source de type "entité morale") ainsi que le prénom/nom s'il s'agit d'une source de type "personne physique"
— Le nom des sources
— Le nom des médias

Ces résultats sont ensuite présentés à l'utilisateur, les entrées sont cliquables :

Le champ texte d'entrée présenté à l'utilisateur est surveillé et toute modification y relative entraîne l'appel de la fonction suivante :

```
$(function() {
  $('#search_terms').on('input', function() {
    var search_terms = $(this).val();

    if (search_terms == "") {
      $('#search_div').html("");
    } else {
      $.get("/reports/search", {"search_terms": search_terms }, function(data) {
        $('#search_div').html(data);
      });
    }
  });
});
```

Le contrôleur "report" (méthode "search") se charge d'effectuer cette recherche :

```
def search
  @search_terms = params[:search_terms]
  @search_results = Hash.new
  @search_results["reports"] = Report.where("title LIKE ? OR head LIKE ? OR text
    LIKE ?", "%#{@search_terms}%", "%#{@search_terms}%", "%#{@search_terms}%").
    order("updated_at DESC").limit(20)
  @search_results["categories"] = Category.where("name LIKE ?", "%#{@search_terms
    }%")
  @search_results["sources"] = Source.where("firstname LIKE ? OR lastname LIKE ? OR
     name LIKE ?", "%#{@search_terms}%", "%#{@search_terms}%", "%#{@search_terms
    }%")
  @search_results["medias"] = Medium.where("name LIKE ?", "%#{@search_terms}%")

  render :partial => "reports_search"
end
```

Une autre implémentation Ajax implémentée est le fait qu'il est possible d'ajouter une source directement dans le formulaire d'ajout/d'édition de news. Lors de l'ajout de la source, celle-ci est automatiquement ajoutée au formulaire d'ajout/d'édition de news. De cette façon, l'utilisateur n'a pas à rechercher la source qu'îl vient d'ajouter, car elle lui est automatiquement proposée.

10 Déploiement

La procédure suivante permet de déployer l'application et de la tester. Nous utilisons les fixtures pour permettre de charger directement la base de données avec des données de test.

- Créer la base de données nommée "news"
- Exécuter :
  `bundle install`
- Exécuter :
  `rake db:migrate`
- Exécuter :
  `rake db:fixtures:load`
- Exécuter :
  `rails s`

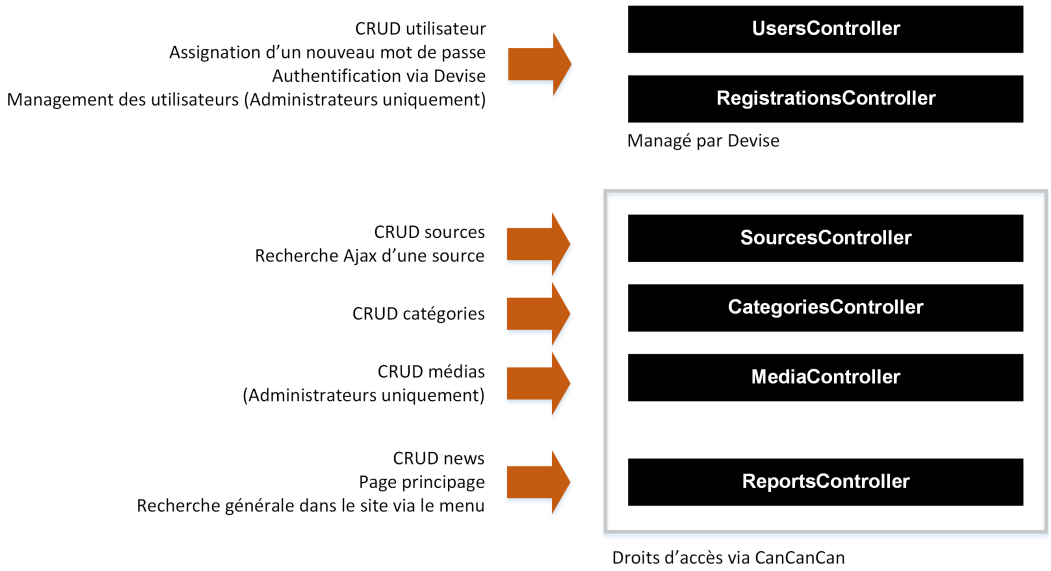
Les comptes créés par les fixtures sont les suivants (email password) :

- admin : jean@paul.com
  `tn^VQBaw`
- auteur : albert@haller.com
  `48n6dqBe`
- auteur : jaime@heig.com
  `gJWOp74`

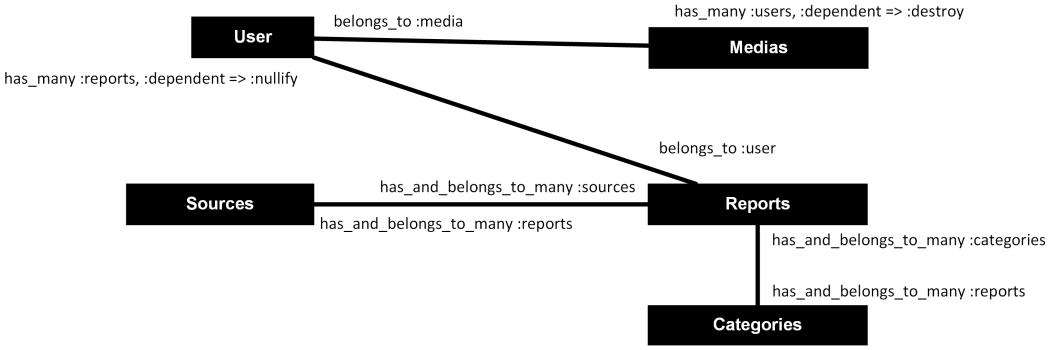
11 MVC

Le mécanisme MVC utilisé est celui fourni par Ruby on Rails.

Les contrôleurs implémentés sont les suivants :



Les relations ORM propres à Rails implémentées sont les suivantes :



12 Librairies utilisées

Les librairies suivantes ont été utilisées dans le projet :

- Bootstrap, pour les styles graphiques
- Devise, pour l'authentification
- CanCanCan, pour la gestion des droits d'accès

Turbolinks a été désactivé, car le Javascript n'était pas correctement exécuté lorsque la page était chargée et il fallait rafraîchir la page pour que le Javascript soit enfin chargé.

13 Etat des lieux

Le projet est fonctionnel et utilisable. Il répond au cahier des charges.

14 Conclusion

Ce projet a été une bonne prise en main de RoR. Nous avons pu consulter ses forces mais également ses faiblesses. Ruby est agréable à utiliser car il permet la génération rapide de squelettes, mais cet avantage a ses limites et un inconvénient majeur est de devoir revoir chaque interface pour y ajouter les relations manquantes. Au niveau organisation, il n'a pas toujours été facile de trouver le temps de travailler sur le projet, compte tenu de notre charge de travail tout au long de ce semestre. En bref, ce projet a été fort utile pour appuyer et mettre en oeuvre ce que nous avons appris durant le cours.

Annexes

A Manuel utilisateur