Polytech'Nice-Sophia
Département Sciences Informatiques

Rapport de Projet

Projet n°20

Tuteur de projet : Peter SANDER
Année universitaire 2006/2007

Maximilien PERRIN – Camille ROUX – Matthieu SEGRET – Mathieu SIVADE

SOMMAIRE

INTRODUCTION

Nous attendions ces trois semaines depuis longtemps. Effectivement, elles représentaient la possibilité d'acquérir de nombreuses connaissances dans le domaine de notre choix. Le monde du web est en train de subir une véritable métamorphose. En effet, on voit apparaître depuis quelques années des applications web très dynamiques et interactives.

Tout a probablement commencé avec l'arrivé des Wikis. Depuis leur apparition, le visiteur n'est plus un simple spectateur, il participe réellement à l'évolution du site Internet. De ce fait, le contenu de ces sites est plus riche, plus complet et plus rapidement mis à jour. Ainsi, l'un des meilleurs exemples de cette évolution est Wikipédia qui est une encyclopédie qui comporte plus de 100 000 articles dans une dizaine de langues.

Puis vint l'AJAX. Grâce à cette technologie, on peut rafraichir seulement une partie d'une page. Les applications utilisant l'AJAX apparaissent donc beaucoup plus rapides et interactives que des sites web classiques. Ceci est appelé le « Web 2.0 ». Les grands acteurs de cette révolution sont :

De nos jours, on parle d'application web (ou "Web App"). Le principe de ces sites est de proposer des services qui étaient jusqu'à présent proposés uniquement par des applications locales, tels que le traitement de texte (Google Docs), le tableur (Google Spreadsheet), la retouche d'image (Fauxto), ...

Les applications web sont de plus en plus populaires car il suffit d'un simple navigateur pour les utiliser, plus besoin d'installer l'application sur tous les ordinateurs ou l'on souhaite s'en servir. De plus, elles permettent la plupart du temps de sauver les documents sur lesquels on travaille en ligne et même parfois d'y travailler dessus à plusieurs et de voir toutes les modifications en temps réel. Tous les grands acteurs du monde de l'informatique s'y préparent (Microsoft, Google, Adobe, ...).

Nous ne pouvions rester de simples spectateurs de cette révolution du cybermonde.

PROBLEMATIQUE

Notre première idée pour un sujet a été de créer une application de travail collaboratif ouverte, c'est-à-dire permettant un grand nombre d'interactions et de partages. En particulier, autorisant le partage de graphiques, d'images, etc. entre plusieurs utilisateurs en temps réel. Au cours de spécifications plus précises, nous sommes arrivés à un affichage de plusieurs objets indépendants, qui sont au final devenus des post-it®, appelés notes.

Concernant le support, le choix a été rapide. Nous sommes tous intéressés par le web, et ce genre d'application correspond exactement au courant web 2.0. Nous avons donc décidé de mettre en place une application web, utilisable depuis un simple navigateur.

Afin de concilier les aspects techniques d'application répartie et de site web, nous nous sommes orientés sur un outil de développement web récent et de plsu en plus populaire, Ruby on Rails. C'est un outil proche de Turbogears que nous avons utilisé lors du cours de Claw. Cette similarité nous a intéressé, car elle nous permettait de comparer ces deux frameworks.

Le sujet final était donc de produire une application web permettant de partager des bureaux entre plusieurs utilisateurs, et d'y afficher des notes visibles et modifiables par tous. Cette application devait répondre à un certain nombre de contraintes : être utilisable par tous (indépendamment de la plate-forme et du navigateur), permettre différents types de contenus (texte mis en page, images…), être robuste, et surtout permettre l'utilisation simultanée par différents utilisateurs.

COOPNOTE

APPLICATIONS CONCURRENTES

D'autres applications proches existaient déjà, avec certaines fonctionnalités similaires. Mais aucune ne correspondait à nos attentes ni ne remplissait les critères principaux.

GOOGLE NOTEBOOK

S'il est possible de travailler en collaboratif, il n'y a pas d'affichage en temps réel. Une sauvegarde et un rafraîchissement restent nécessaires pour accéder au travail des autres utilisateurs.

De plus, les fonctionnalités sont très restreintes, n'autorisant que des blocs superposés, et que du texte (avec une mise en forme minimale).

WEB NOTE

Application sans doute la plus proche de notre sujet, elle permet d'afficher des post-it et de les partager. Néanmoins, ce partage est désynchronisé (nécessite une sauvegarde et un rafraîchissement), et les bureaux ne permettent qu'un seul utilisateur simultanément.

De plus, ce site ne fournit aucune sécurisation : tous les bureaux sont accessibles et modifiables, et le contenu n'est pas restreint, autorisant le html, les scripts, etc.

APPLICATIONS LOCALES (TOMBOY, ETC.)

Il existe de nombreuses applications locales (installées sur la machine) permettant de créer des notes. Elles sont néanmoins restreintes par natures ; elles ne permettent en effet pas de partager ces notes, de les utiliser sur une autre machine, etc.

La plupart sont également restreintes à du texte simple, qui, s'il suffit à l'utilisation, ne correspond pas à nos objectifs.

UNICITE DU PROJET

Notre projet se distingue des applications proches par ces différences. Mais la principale particularité reste l'affichage des modifications en temps réel, et non pas à des intervalles fixes. Chaque modification est immédiatement répercutée chez tous les utilisateurs. C'est une fonctionnalité qui n'est pour l'instant présente dans aucun autre logiciel.

CHOIX DES SOLUTIONS

CHOIX DU LANGAGE & FRAMEWORK

Afin de nous donner les meilleurs outils pour parvenir à atteindre les objectifs de notre projet, nous avons comparé les principaux langages (et frameworks associés) permettant le développement d'applications web.

PHP & CODE IGNITER

PHP offre une excellente portabilité (très peu d'hébergeurs ne le proposent pas), une rapidité de développement raisonnable, et un excellent support de sa communauté (documentation, forums, etc). De plus, nous nous étions déjà formés à Code Igniter lors du projet de Janvier.

PYTHON & TURBOGEARS

Après le projet de CLAW, Turbogears nous avait déçu : documentation incomplète, approche pas toujours intuitive, portabilité très restreinte, possibilités offertes par le framework réduites. La rapidité de développement est cependant intéressante.

RUBY & RUBY ON RAILS

Certains d'entre nous ayant expérimenté RoR à titre personnel ont pu nous rapporter la facilité d'utilisation, l'excellente documentation, bien que la portabilité soit largement moindre que celle de PHP. Cependant, la rapidité de développement est aussi très importante avec ce langage.

Nous avons donc placé notre choix sur Ruby on Rails : le peu de temps dont nous disposions pour ce projet, conjugué à notre volonté d'élargir nos connaissances des technologies de développement du web 2.0, font de Ruby accompagné de RoR le choix le plus adapté, malgré le faible nombre d'hébergeurs supportant ce langage.

CHOIX DU SYSTEME DE MISE A JOUR AUTOMATIQUE DES PAGES

Ensuite, nous devions choisir un mode de mise à jour des pages pour assurer la synchronisation des utilisateurs. Pour cela, nous avions le choix entre deux technologies : AJAX et Comet.

AJAX

AJAX permettrait une mise à jour périodique des pages : selon la période de rafraichissement choisie, on peut dans un extrême rafraichir constamment et user de la bande passante inutilement, ou dans l'autre extrême rafraichir peu souvent et ne pas voir les modifications effectuées par les autres utilisateurs en "temps réel".

COMET

Comet permet quand à lui de maintenir une connexion persistante au serveur afin de tenir les clients au courant des mises à jour dès qu'une modification a lieu. Ainsi, la bande passante utilisée est minimale, la mise à jour aussi rapide que possible.

Comet ne comporte donc que des avantages du point de vue technologique, pour notre application, cependant les outils Ruby pour l'utiliser sont plutôt rares. Nous en avons expérimenté deux.

CHOIX DU PLUGIN COMET

JUGGERNAUT

Ce plugin repose sur plusieurs technologies pour établir la connexion persistante à la manière de Comet. Cependant, ce n'est pas exactement du Comet car il repose sur des technologies propriétaires comme Flash.

Lorsque nous avons commencé le projet, Juggernaut utilisait Flash en version 6, qui est installé chez 95% des utilisateurs d'internet. Il semblait donc présenter une excellente compatibilité et nous avons essayé de l'utiliser. Mais assez vite, nous nous sommes rendu compte qu'il n'était pas compatible avec Internet Explorer, que de nombreux bugs étaient présents, et que la compatibilité avec Flash 6 n'était en fait pas bien assurée, même sur des ordinateurs le possédant.

Nous avons donc délaissé ce plugin qui nous avait paru à première vue prometteur, pour nous pencher sur...

SHOOTING STAR

Ce plugin, en version alpha (ce qui veut dire qu'il peut lui manquer des fonctionnalités), documenté principalement en japonais, nous avait au premier abord paru un peu aventureux. Mais un tutorial en anglais suffisamment complet nous a permis de le mettre en place pour nous rendre compte qu'il était beaucoup plus compatible que Juggernaut et qu'il implémentait vraiment Comet ; c'est à dire qu'il n'utilise que du Javascript et pas de technologies propriétaires contraignantes. C'est finalement ce plugin qui a été intégré au projet, et nous satisfait entièrement.

RESULTATS

FONCTIONNALITES

On peut créer, éditer et partager un bureau. Celui-ci peut contenir des notes que l'on peut éditer, déplacer, redimensionner ou coloriser.

STRUCTURE MVC

Nous nous sommes reposés sur l'architecture MVC (Modèle-Vue-Contrôleur) proposée par Rails pour développer l'interface web. Cette structure permet une séparation entre le traitement de l'information (modèles), l'interface utilisateur (vues) et la synchronisation pour mettre à jour la vue ou le modèle (contrôleurs).

DEVELOPPEMENT

Nous avons adopté un développement incrémental, qui consiste à développer l'application fonctionnalité par fonctionnalité. Rails facilite beaucoup cette méthode de développement à l'aide de son système de migration de base de données. En effet celui-ci permet de garder en mémoire les différentes versions du schéma (on peut voir cela comme les incréments qui font passer le schéma d'une version à une autre).

Cette méthode permet de fournir rapidement une version de l'application proposant uniquement les fonctionnalités déjà développées.

MODELES

Chaque modèle sert donc d'interface avec la table correspondante dans la base de données. Nous avons utilisé le système Active Record de Rails de type ORM, qui fournit l'illusion d'une base de données orientée objet en cachant le coté relationnel de celle-ci. Ce système permet de garantir une certaine indépendance vis-à-vis du SGBD (MySQL, Postgres, etc.) et de manipuler les tables de manière plus naturelle.

- Une note (Note) possède une position, une taille, un contenu et l'une des 5 couleurs disponible. Le zindex permet de mémoriser la superposition des notes.
- Un bureau (Desktop) possède un nom, une visibilité (public ou privé). Un bureau possède des notes.
- Un utilisateur (User) possède un login et un email unique, le hash code du mot de passe et une clé permettant le cryptage du mot de passe (salt), ainsi que la date d'expiration du cookie, la date de création et de mise à jour du profil utilisateur. Un utilisateur peut posséder plusieurs bureaux et un bureau peut posséder plusieurs utilisateurs. Cette relation N-N est possible grâce à la table de jointure Right.
- Un droit (Right) a pour rôle de définir un droit d'accès pour un utilisateur par rapport à un bureau. De plus le champ position a été ajouté par la suite afin de définir un ordre à la disposition des bureaux pour un utilisateur. Ainsi la table droit ne porte plus très bien son nom, bien que son utilité première consiste en la gestion des droits.
- Un tag (Tag) a un nom. Celui ci possède une relation n-n avec la table Desktop.
- Meteor permet de sauvegarder le JavaScript envoyé au client. En cas de perte de la connexion avec le client (dû à un timeout par exemple), MeteorStrike va chercher dans cette table les scripts qu'il n'a pu envoyer et les envoie au client. Le nombre de tuples dans cette table est vite important, il est donc nécessaire de la vidé régulièrement (grâce à un cron sur le serveur par exemple).

AUTHENTIFICATION

Nous avons utilisé le plugin Acts_as_Authenticated, qui gère l'inscription et l'authentification des utilisateurs. La table User est ainsi automatiquement créée avec des champs par défaut (login, email, crypted_password, salt, created_at, updated_at, remember_token, remember_token_expires_at). Le mot de passe est ainsi crypté à l'aide de la clé salt. Le "token" permet d'identifier un utilisateur et lui donner accès à sa session, celui-ci est stocké dans un cookie.

TESTS

Nous avons tenue une attention particulière aux tests de notre application. En effet, le ratio entre nos lignes de code de test et nos lignes de code de l'application elle-même est de 0.8.

Nous avons ainsi utilisé des tests unitaires pour vérifier le comportement de chacune des méthodes du modèle. Ces derniers permettent de vérifier que le modèle refuse bien les données non-valides par exemple.

Des tests fonctionnels ont aussi été effectués. Ces tests se situent à un plus haut niveau. Ils permettent de garantir que chacune des actions effectuées à la suite d'une requête donne le comportement souhaité (redirection, affichage d'une information précise, ...).

QUELQUES STATISTIQUES SUR L'APPLICATION

Name	Lines	Lines Of Code	Classes	Methods	M/C	LOC/M
Controllers	526	453	5	40	8	9
Helpers	11	10	0	0	0	0
Models	230	169	7	22	3	5
Libraries	233	145	3	30	10	2
Integration tests	0	0	0	0	0	0
Functional tests	566	452	10	74	7	4
Unit tests	171	140	6	23	4	4
Total	1737	1369	31	189	6	5

Code LOC (Lines Of Code): 777 Test LOC: 592 Code to Test Ratio: 1:0.8

POINTS A RAJOUTER

AUGMENTER LA SECURITE

Il est tout d'abord important d'ajouter une interface administrateur et modérateur pour surveiller et contrôler le contenu des notes. Et éventuellement avoir le pourvoir de bannir un utilisateur.

Par la suite l'ajout de message d'acceptation à une invitation paraît primordial. Cela pour éviter qu'un utilisateur mal intentionné ajoute tous les utilisateurs à un bureau contenant du spam, dans l'état actuel de l'application le bureau du spammeur apparaît dans la liste de bureaux des autres utilisateurs sans leur consentement.

Enfin il pourrait être utile d'ajouter un plugin de type Captcha, permettant de limiter les inscriptions automatisées.

FACILITER LE TRAVAIL COOPERATIF

Il serait intéressant de notifier l'arriver d'un utilisateur écrivain aux autres utilisateurs possédant un statut permettant l'édition et connectés sur le même bureau.

Une autre fonction très utile serait d'informer les utilisateurs possédant le droit d'édition, d'éventuelles modifications en cours sur une note.

Enfin l'ajout de quelques informations sur le profil utilisateur permettrait la personnalisation de l'application et la création d'un réseau social (nom, prénom, projet en cours, couleur par défaut etc.).

ACCESSIBILITE

Un ordonnancement automatique des notes améliorerait l'ergonomie de l'application.

De plus, l'ajout de notes personnalisées de type vidéo, photo, flux RSS permettrait une utilisation plus variée des notes.

L'ergonomie se trouverait renforcée si le transfert/la copie de notes entre bureaux serait possible.

D'autres améliorations de l'accessibilité seraient les nuages de tags, l'internationalisation de l'application et l'ajout d'aide en ligne.

BILAN

L'application correspond à ce que nous voulions: un outil de travail collaboratif simple, permettant des utilisations dérivées variées.

Nous avons été très étonnés de la rapidité de mise en œuvre au début du développement. En effet, Ruby on Rails est un outil extrêmement puissant pour la réalisation des parties classiques d'un site web.

Il reste cependant quelques fonctionnalités à ajouter si l'on souhaite mettre **CoopNote** en production.

CONCLUSION

Notre vœu était de parfaire nos connaissances dans les technologies du web et nous sommes tous d'accord pour dire qu'il a été exhaussé. En effet, ce projet a été tout d'abord l'occasion de concevoir un site web dans sa globalité et d'utiliser des technologies très récentes. Nous sommes à présent familiers avec le Ruby, le CSS2 ou encore le JavaScript. De plus, nous comprenons maintenant le fonctionnement des sites qui révolutionnent actuellement le web et avons appris à faire des sites dynamiques et interactifs grâce à l'AJAX et Comet.

La réalisation de **CoopNote** nous a également permis d'approfondir notre utilisation de l'architecture MVC. Celui-ci nous a aidé à bien séparer les tâches et à mieux structurer le code. Il nous serait difficile de concevoir un site web avec une architecture différente tellement celle-ci nous semble évidente maintenant.

Pour finir, ce projet a été une expérience enrichissante sur le point humain. Ce n'était pas la première fois que nous travaillions sur un projet commun, mais nous avons tout de même apprécié réfléchir à la manière de gérer le temps et d'utiliser au mieux les compétences de chacun afin de d'atteindre les objectifs que nous nous étions fixés.

Nous attendons tous avec impatience de renouveler une expérience de ce type, à travers le stage ou les projets de troisième année.

NOTICE D'UTILISATION

INSTALLATION DE COOPNOTE

PREREQUIS

- Système Linux
- Ruby 1.8.6+
- Ruby on Rails 1.2.3+
- RubyGem 0.9.4+

INSTALLATION DES GEMS

COMMENT INSTALLER UN GEM

Une fois les prérequis correctement installés sur votre système, il suffit d'utiliser la commande suivante pour installer un gem:

```
gem install [nom du gems] -y
```

LISTE DES GEMS NECESSAIRES

CoopNote a besoin que certains gems soient installés. Voici la liste:

- actionmailer (1.3.3)
- actionpack (1.13.3)
- activerecord (1.15.3)
- activesupport (1.4.2)
- acts_as_taggable (2.0.2)
  An acts-as Mixin for easy applying and searching tags/folksnomies on Active Record objects
- ferret (0.11.4)
  Ruby indexing library.
- mongrel (1.0.1)
  A small fast HTTP library and server that runs Rails, Camping, Nitro and Iowa apps.
- rails (1.2.3)
  Web-application framework with template engine, control-flow layer, and ORM.
- rake (0.7.3)
  Ruby based make-like utility.
- RedCloth (3.0.4)
  RedCloth is a module for using Textile and Markdown in Ruby. Textile and Markdown are text formats. A very simple text format. Another stab at making readable text that can be converted to HTML.
- shooting_star (2.0.1)
  Our goal is development of practical comet server which will be achieving over 100,000 simultaneous connections per host. On this purpose, we abandon portability and use system calls depending on particular OS such as epoll and kqueue.

CREATION ET CONFIGURATION DE LA BASE DE DONNEE

CREATION DE LA BASE

Pour la base, vous avez le choix. Un grand nombre de type de base est supporté (MySQL est tout de même, la base la plus utilisée par la communauté Rails). Vous devez donc créer une base vide avec un utilisateur ayant les droits suffisants.

CONFIGURATION

Vous devez, à présent, mettre les paramètres de votre base de données dans le fichier suivant :
`./config/database.yml` (partie production)

CREATION DES TABLES

La création des tables est une simple formalité avec Rails, grâce aux migrations. Il suffit d'entrer la commande suivante:

```
RAILS_ENV=production rake migrate
```

CONFIGURATION DU SERVEUR DE PUSH

Pour configurer le serveur de push, vous devez connaitre votre IP internet si vous souhaitez mettre le site à disposition sur le net, sinon une adresse IP locale suffit.
Vous devez modifier l'adresse du serveur de push dans les fichiers suivant

- `./config/database.yml` :
  common: &common
  shooting_star: # URI to connect from brower to comet server
  server:
  – :8080

- `./config/shooting_star.yml` :
  server:
  host: 157.169.100.193
  port: 8080

LANCEMENT DU SERVER

SERVEUR DE BASE

Il faut tout d'abord démarrer le serveur de push (permettant l'utilisation de la technologie Comet):

```
shooting_star start
```

Par defaut, Rails propose un serveur écrit en Ruby appelé **Webrick**.
Pour le lancer il suffit de se placer à la racine du projet et de taper la commande suivante:

```
RAILS_ENV=production .script/server
```

UTILISATION DE MONGREL

Mongrel plus performant que Webrick. Il est souvent utilisé dans des environnements de production. Pour l'utiliser il est nécessaire d'installer le gems `mongrel`.
Voici la commande à taper à la racine du projet pour l'utiliser:

```
RAILS_ENV=production sudo mongrel_rails start -p [port]
```

REPARTITION DE CHARGE AVEC APACHE

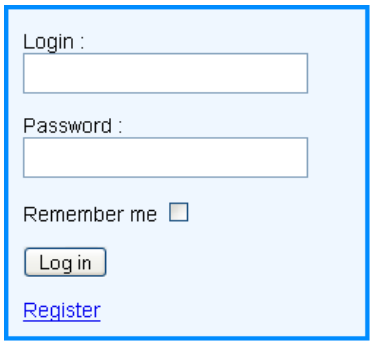
Afin d'optimiser la charge du serveur, il est possible de lancer plusieurs instances de Mongrel et de se servir d'Apache en frontal pour répartir la charge. Vous pourrez trouver plus de details sur le site suivant:
http://mongrel.rubyforge.org/docs/apache.html

LES FONCTIONNALITES OFFERTES A L'UTILISATEUR

AUTHENTIFICATION

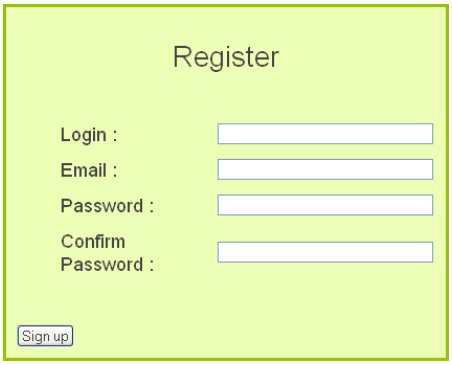
LOGIN

Interface d'authentification de la page d'accueil. Possibilité de mémoriser la session pour les visites ultérieures.



ENREGISTREMENT

Interface d'enregistrement d'un nouvel utilisateur. La création du compte est immédiate, mais il est possible de mettre très rapidement en place un système de sécurisation par envoi de mail.



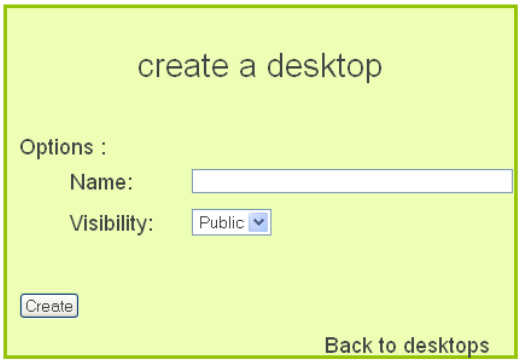
BUREAU

CREER

Pour créer un bureau, il suffit de cliquer sur l'onglet `create desktop` 

qui ouvre un formulaire simple avec deux champs:

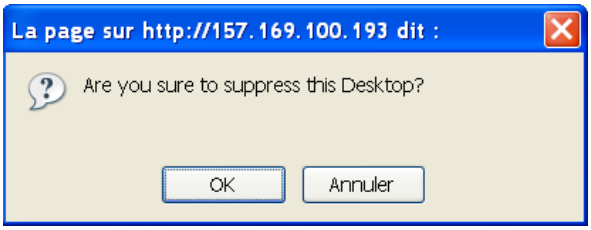
- **Name** : Le nom du bureau qu'il est possible de modifier ultérieurement.
- **Visibility** : Ce champ permet de définir si le bureau est visible aux personnes non invitées sur le bureau. Ce paramètre est aussi modifiable par la suite.



SUPPRIMER

Pour supprimer un onglet, il suffit de cliquer sur la croix sur l'onglet du bureau en question 

. Ensuite, une boite de dialogue s'ouvrira pour demander confirmation afin d'éviter toute mauvaise manipulation.



PREFERENCES

Il est possible de changer à tout moment le nom du bureau et sa visibilité grâce au lien `Preferences` de la barre située sous les onglets.

CHANGER

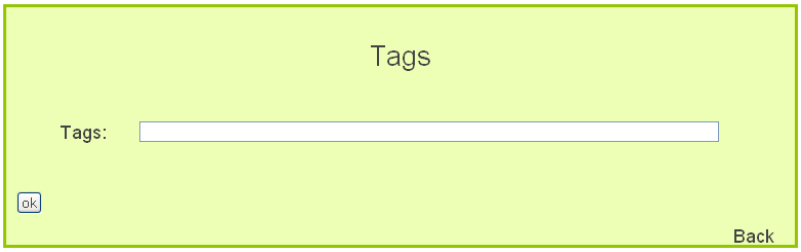
Pour changer de bureau courant, il suffit de cliquer sur l'onglet correspondant.

DEPLACER

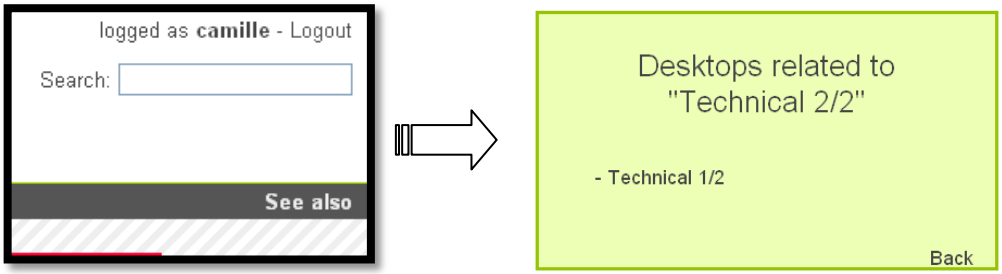
Il possible de changer l'ordre des bureaux de son espace personnel par Drag and Drop (glisser-déplacer). Chaque onglet peut être attrapé puis déplacé par son bord gauche.

TAGS

Il est possible de tagger chaque bureau en donnant une liste de mots séparés par des espaces.



A partir de ces mots-clés, **CoopNote** est capable de donner la liste des bureaux qui possèdent le plus de mots-clés avec le bureau courant. Pour voir cette liste, il suffit de cliquer sur le lien `see also` (en haut, à droite).



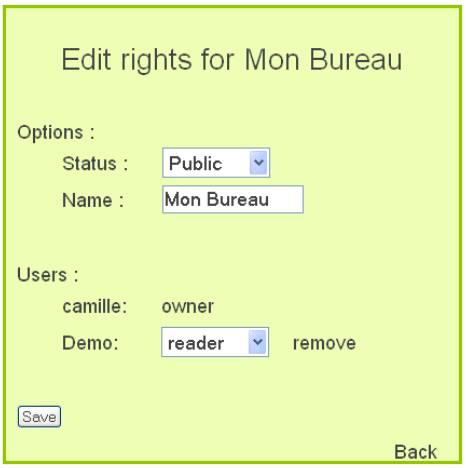
MODIFIER LES DROITS

Il est possible de définir des droits d'accès pour filtrer l'accessibilité à un bureau. Afin de permettre par exemple à votre groupe de projet de travailler sur le bureau que vous avez créé à cette occasion, vous pouvez utiliser la fonctionnalité d'invitation (`Invite`, en haut à gauche).

Lors de l'invitation, vous avez le choix entre plusieurs niveaux de droits à donner à la personne invitée :

- 'reader' lui permettra de voir l'évolution de votre bureau en temps réel, mais il ne pourra y participer en aucun cas (comme s'il regardait une vidéo de votre bureau). Les personnes que vous voulez informer de votre travail pourront ainsi voir son évolution.
- 'writer' lui permettra d'écrire lui aussi des notes, les modifier, les supprimer. Votre tuteur de projet pourra ainsi donner des idées, proposer des corrections, etc.
- 'owner' donnera tout les droits sur le bureau : il pourra donc en changer le nom et les tags, inviter des personnes à le rejoindre, et éditer les préférences au même titre que vous. Vos collègues de projet apprécieront de travailler de pair avec vous.

Une fois un bureau créé, vous pouvez en modifier plusieurs propriétés grâce au lien `Preferences` (en haut à gauche, à droite de `Invite`).



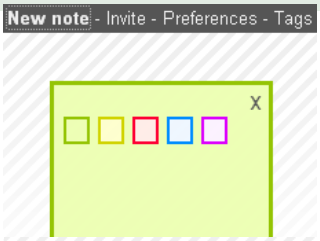
Il y est possible de modifier le nom du bureau, sa visibilité (`public` ou `private`), et les droits des utilisateurs qui ne sont pas 'owner'. De cette manière, un « propriétaire » du bureau ne le quitte que de sa propre volonté.

RECHERCHE

Il est possible d'effectuer une recherche plein texte sur les notes pour trouver les bureaux contenants les informations recherchées. Ceci s'effectue via le champ de recherche en haut à droite :

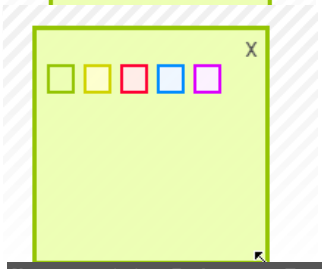
Après y avoir inséré les mots-clés de recherche et appuyé sur entrée, vous obtiendrez la page des résultats :

NOTES



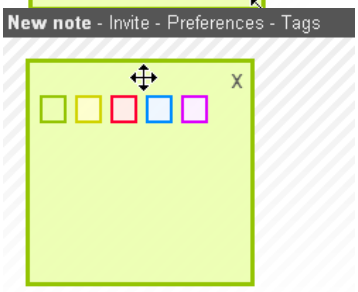
CREATION

Une nouvelle note peut être créée soit en cliquant sur le lien approprié, soit en double-cliquant sur le bureau. Le lien crée la fenêtre dans le coin haut gauche, le double-clique à l'emplacement du curseur.



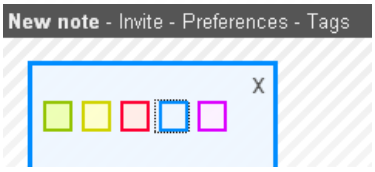
REDIMENSIONNEMENT

La taille d'une note peut être modifiée simplement en tirant sur les bordures droite et/ou bas.



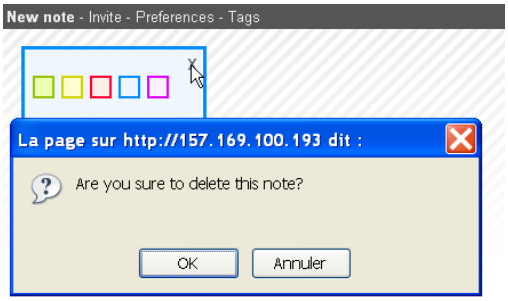
DEPLACEMENT

Une note peut être déplacée simplement par glisser-déposer, en cliquant sur sa partie supérieure.



CHANGEMENT DE COULEUR

La couleur peut être changée simplement en cliquant sur l'une des configurations disponibles.



SUPPRESSION

Une note peut être supprimée en cliquant sur la croix en haut à droite. Un message est affiché pour demander confirmation de la suppression.

EDITION DE CONTENUS RICHES

CoopNote permet d'entrer du contenu formaté dans les notes grâce à une syntaxe de plus en plus utilisé, Textile (voir Annexe).

En plus de la syntaxe Textile, **CoopNote** transforme automatiquement les URLs en lien du même nom. *Les balises HTML sont interdites afin d'éviter l'insertion de code malveillants dans les notes.*

Voici quelques exemples:

- _italique_ => *italique*
- *gras* => **gras**
- -barré- => ~~barré~~

SCENARIOS D'UTILISATION

ENREGISTREMENT

L'utilisateur arrive sur la page d'accueil du site. Il clique sur "s'enregistrer" et arrive sur la page d'enregistrement. Il remplit les champs (nom, adresse mail, mot de passe, confirmation) et accepte. Il est redirigé sur la page d'accueil.

CONSULTATION PUBLIQUE

L'utilisateur connaît l'adresse d'un bureau publique. Il va à cette adresse, et accède au bureau en question.

CONSULTATION PRIVEE

L'utilisateur arrive sur la page d'accueil. Il rentre son nom et son mot de passe, et est redirigé vers son dernier bureau visité. Il clique sur un autre bureau et le consulte. Il clique sur logout et est redirigé sur la page d'accueil.

MODIFICATION

L'utilisateur s'enregistre et accède au bureau désiré. Il modifie le contenu d'une note puis quitte.

CREATION DE NOTES

L'utilisateur s'enregistre et accède au bureau désiré. Il double-clique sur le bureau et crée une note. Il choisit une autre couleur, double-clique sur la note, et rentre un contenu.

CREATION DE BUREAUX

L'utilisateur s'enregistre. Il clique sur "create desktop" et est redirigé sur la page de création de bureau. Il rentre un nom, choisit l'accessibilité du bureau et valide. Il est redirigé sur le bureau précédent. Il clique sur le nouveau bureau, puis sur "invite". Il est redirigé sur la page d'invitation. Il rentre le nom de la personne à inviter, les droits qu'il lui accorde, et valide. Il est redirigé sur le bureau.

GLOSSAIRE

AJAX

Asynchronous JavaScript And ***XML*** (« XML et Javascript asynchrones »), est un acronyme désignant une méthode informatique de développement d'applications Web. AJAX n'est pas une technologie en elle-même, mais un terme qui évoque l'utilisation conjointe d'un ensemble de technologies couramment utilisées sur le Web :

- **HTML** (ou **XHTML**) pour la structure sémantique des informations
- **CSS** pour la présentation des informations
- **DOM** et **JavaScript** pour afficher et interagir dynamiquement avec l'information présentée
- l'objet **XMLHttpRequest** pour manipuler les données de manière asynchrone avec le serveur Web.

FLUX RSS

Un **flux RSS**, sigle de **Really Simple Syndication** (*souscription vraiment simple*), est un format de syndication de contenu Web, codé sous forme XML. Ce système permet de diffuser en temps réel les nouvelles des sites d'information, ce qui permet de rapidement consulter ces dernières sans visiter le site.

FRAMEWORK

Un **framework** est un ensemble de bibliothèques permettant le développement rapide d'applications. Il fournit suffisamment de briques logicielles pour pouvoir produire une application aboutie. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres.

JAVASCRIPT

JavaScript est un langage de programmation de type script, orienté objets à prototype, principalement utilisé dans les pages Web. C'est une des composantes principales de l'Ajax.

LOCALISATION

La **localisation** d'un logiciel concerne le processus de traduction de l'interface utilisateur d'une application d'une langue vers une autre et en l'adaptant à la culture locale. La localisation est souvent désignée sous le terme de l10n.

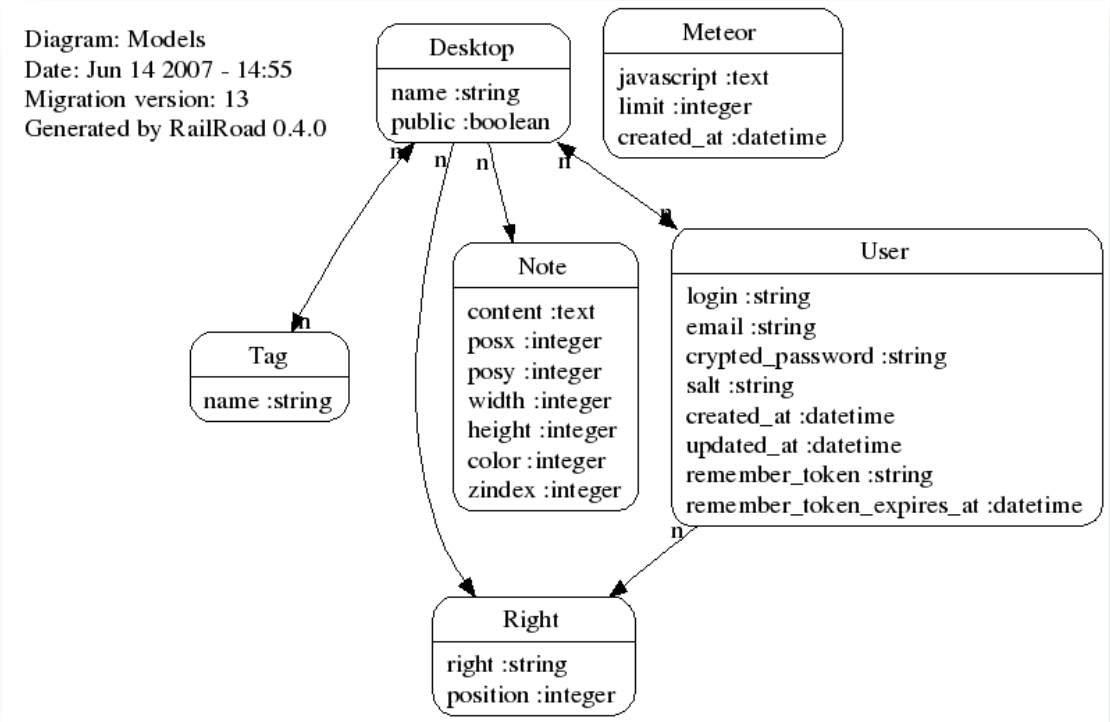
PLUGIN

En informatique, le terme anglais **plugin**, est employé pour désigner un programme qui interagit avec un logiciel principal, appelé *programme hôte*, pour lui apporter de nouvelles fonctionnalités.

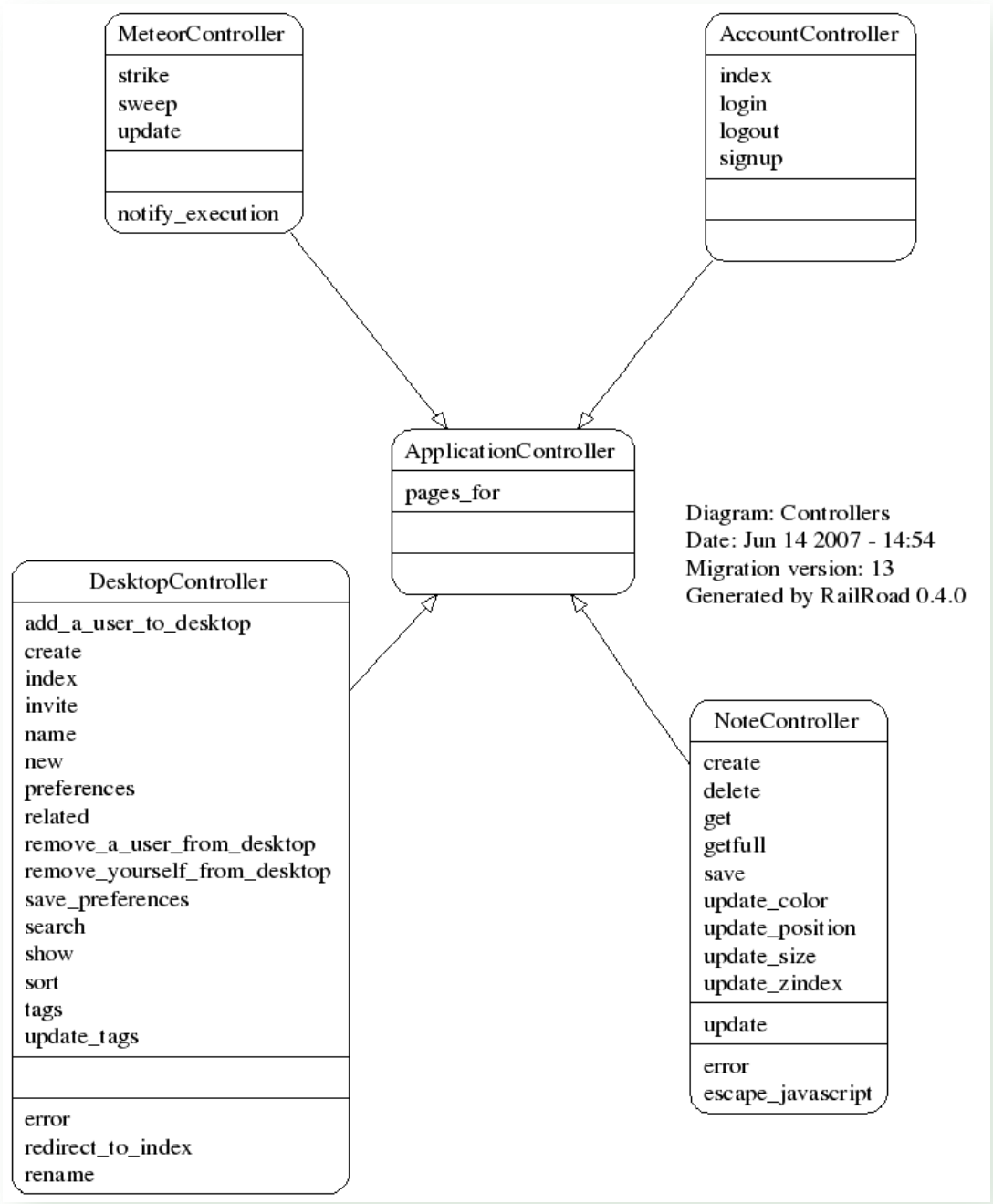
RUBY

Ruby est un langage de programmation orienté objet interprété, créé par Yukihiro Matsumoto, basé sur le principe de moindre surprise. Il doit sa récente popularité au framework Ruby On Rails.

STRUCTURE DES MODELES



STRUCTURE DES CONTROLLEURS



`A single paragraph.`
 	
`Followed by another.`
	→	A single paragraph.
		
		Followed by another.

`I spoke.` `And none replied.`	→	I spoke. And none replied.
`"Observe!"`	→	“Observe!”
`Observe -- very nice!`	→	Observe—very nice!
`Observe - tiny and brief.`	→	Observe – tiny and brief.
`Observe...`	→	Observe…
`Observe: 2 x 2.`	→	Observe: 2×2.
`one(TM), two(R), three(C).`	→	one™, two®, three©.
Quick Block Modifiers		
`h1. Header 1`	→	Header 1
`h2. Header 2`	→	Header 2
`h3. Header 3`	→	Header 3

`An old text`
 	
`bq. A block quotation.`
 	
`Any old text`
	→	An old text
		
		A block quotation.
		
		Any old text

`This is covered elsewhere[1].`	→	This is covered elsewhere[1].
`fn1. Down here, in fact.`	→	[1] Down here, in fact.
Quick Phrase Modifiers		
`I _believe_ every word.`	→	I BELIEVE every word.
`And then? She *fell*!`	→	And then? She **fell**!
`I __know__.` `I **really** __know__.`	→	I *know*. I **really** *know*.
`??Cat's Cradle?? by Vonnegut`	→	*Cat’s Cradle* by Vonnegut
`Convert with @r.to_html@`	→	Convert with `r.to_html`
`I'm -sure- not sure.`	→	I’m ~~sure~~ not sure.
`You are a +pleasant+ child.`	→	You are a pleasant child.
`a ^2^ + b ^2^ = c ^2^`	→	$a^2 + b^2 = c^2$
`log ~2~ x`	→	$\log_2 x$
`I'm %unaware% of most soft drinks.`	→	I’m unaware of most soft drinks.
`I'm %{color:red}unaware%` `of most soft drinks.`	→	I’m unaware of most soft drinks.
Attributes		
`p(example1). An example`	→	An example
`p(#big-red). Red here`	→	Red here
`p(example1#big-red2). Red here`	→	Red here

`p{color:blue;margin:30px}. Spacey blue`	→	Spacey blue
`p[fr]. rouge`	→	rouge
`I seriously *{color:red}blushed* when I _(big)sprouted_ that corn stalk from my %[es]cabeza%.`	→	I seriously **blushed** when I SPROUTED that corn stalk from my cabeza.
`p<. align left`	→	align left
`p>. align right`	→	align right
`p=. centered`	→	centered
`p<>. justified`	→	justified
`p(. left ident 1em`	→	left ident 1em
`p((. left ident 2em`	→	left ident 2em
`p))). right ident 3em`	→	right ident 3em
`h2()>. Bingo.`	→	Bingo.
`h3()>[no]{color:red}. Bingo`	→	Bingo
Lists		

`# A first item`
`# A second item`
`# A third`
	→	1. A first item
		2. A second item
		3. A third

`# Fuel could be:`
`## Coal`
`## Gasoline`
`## Electricity`
`# Humans need only:`
`## Water`
`## Protein`
	→	1. Fuel could be:
		  1. Coal
		  2. Gasoline
		  3. Electricity
		2. Humans need only:
		  1. Water
		  2. Protein

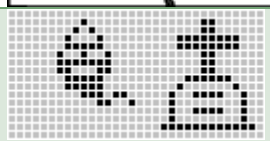
`* A first item`
`* A second item`
`* A third`
	→	• A first item
		• A second item
		• A third

`* Fuel could be:`
`** Coal`
`** Gasoline`
`** Electricity`
`* Humans need only:`
`** Water`
`** Protein`
	→	• Fuel could be:
		  ○ Coal
		  ○ Gasoline
		  ○ Electricity
		• Humans need only:
		  ○ Water
		  ○ Protein

External References		
`I searched "Google":http://google.com.`	→	I searched Google.

`I am crazy about "Hobix":hobix and "it's":hobix "all":hobix I ever "link to":hobix!`
 	
`[hobix]http://hobix.com`
	→	I am crazy about Hobix and it's all I ever link to!

`!http://hobix.com/sample.jpg!`	→	
`!openwindow1.gif(Bunny.)!`	→	
`!openwindow1.gif!:http://hobix.com/`	→	

`!>obake.gif!`
 	
`And others sat all round the small machine and paid it to sing to them.`
	→	
		
		And others sat all round the small machine and paid it to sing to them.

`We use CSS(Cascading Style Sheets).`	→	We use CSS.
Tables		

`| name | age | sex |`
`| joan | 24 | f |`
`| archie | 29 | m |`
`| bella | 45 | f |`
	→	name age sex
		
		joan 24 f
		
		archie 29 m
		
		bella 45 f

`|_. name |_. age |_. sex |`
`| joan | 24 | f |`
`| archie | 29 | m |`
`| bella | 45 | f |`
	→	**name** **age** **sex**
		joan 24 f
		archie 29 m
		bella 45 f

`|_. attribute list |`
`|<. align left |`
`|>. align right|`
`|=. center |`
`|<>. justify |`
`|^. valign top |`
`|~. bottom |`
	→	**attribute list**
		
		align left
		
		align right
		
		center
		
		justify
		
		valign top
		
		bottom

`|\2. spans two cols |` `| col 1 | col 2 |`	→	spans two cols col 1 col 2

`|/3. spans 3 rows | a |`
`| b |`
`| c |`
	→	spans 3 rows a

		b c
`|{background:#ddd}. Grey cell|`	→	Grey cell

`table{border:1px solid black}.`
`|This|is|a|row|`
`|This|is|a|row|`
	→	This is a row
		This is a row

`|This|is|a|row|`
`{background:#ddd}.`
`|This|is|grey|row|`
	→	This is a row
		This is grey row