Rapport du projet tutoré "Tsunami"

par Renaudeau Gaëtan et Namolovan Nicolae

Sommaire

Introduction

Notre projet est une application web de communication et collaboration en temps réel appelée Tsunami.

Tsunami est destiné à un groupe de personnes qui veulent éditer un même document au même instant. Chaque modification faite par chacun est instantanément visible pour tous ce qui a pour but d'améliorer la collaboration et la productivité du groupe.
Afin de parvenir à nos fins, nous avions de nombreux défis à relever. En effet le web, à ses débuts, n'était pas prévu pour réaliser des applications aussi dynamiques. Bien que les technologies se sont nettement améliorés, certains points ne sont pas encore parfait. Notre application repose donc sur de nombreuses briques logiciels offrant un couche suffisamment intéressante pour se concentrer sur l'essentiel tout en apportant une meilleure compatibilité.

Fonctionnalités de l'application

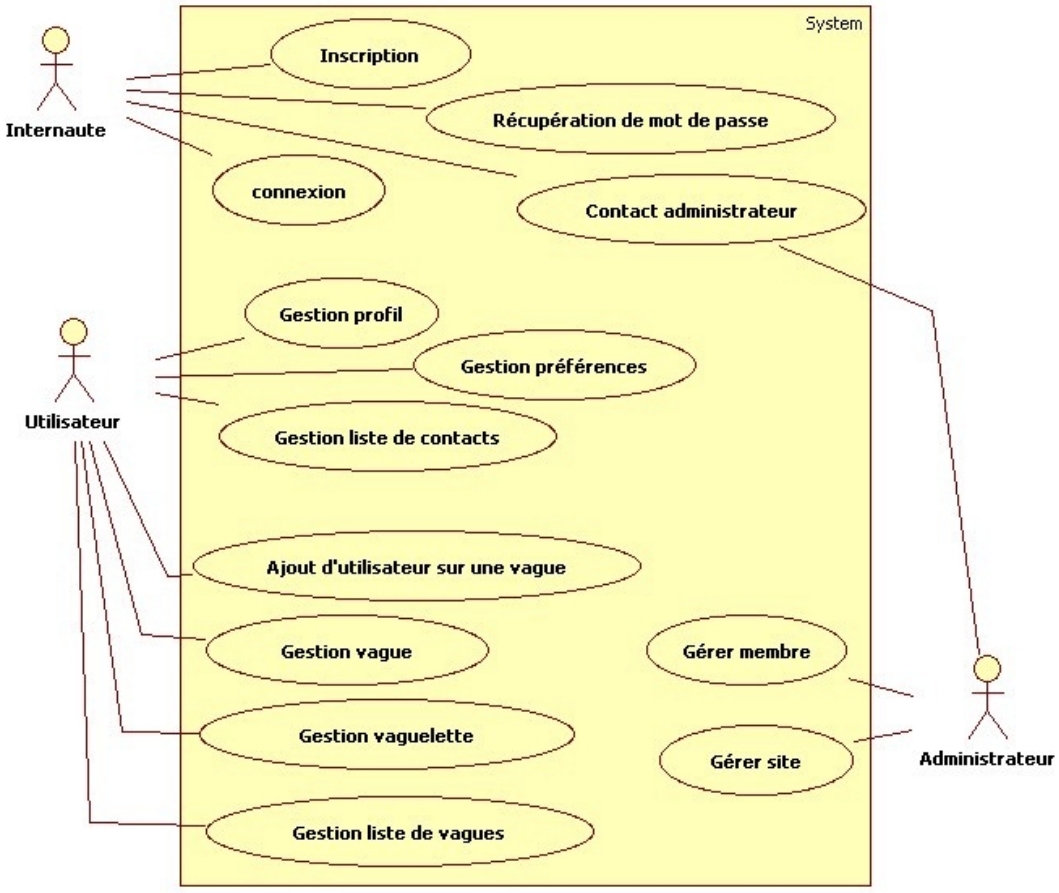
Choix de l'étude de cas

Tsunami est une application web collaborative permettant de suivre et de participer à des discussions entre plusieurs utilisateurs.

Expression initiale des besoins

L'application réunit les concepts du mail, du forum et de la messagerie instantanée.
Ce système est un outil collaboratif permettant à des utilisateurs de suivre des discussions à plusieurs en même temps (c'est à dire de façon instantanée).
Chaque utilisateur peut créer une discussion et y inviter de nouveaux participants au moyen de sa liste de contacts.
La particularité de notre application est que plusieurs utilisateurs peuvent travailler au même instant sur une même discussion.

Diagramme de cas d'utilisation



Vision du projet

Positionnement

- Proposer un outil collaboratif pour un groupe de personnes : une équipe, un groupe d'amis, ...
- Proposer une solution alternative à Google Wave

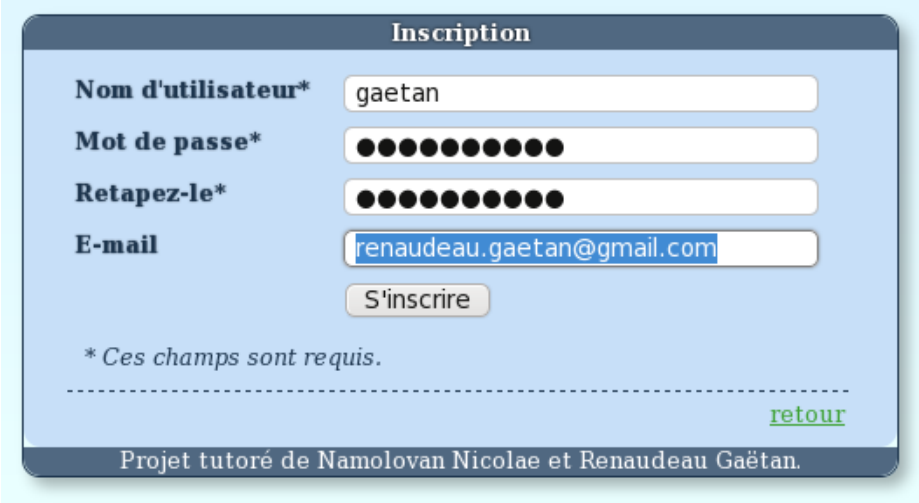
Exigences fonctionnelles

Inscription / connexion / déconnexion d'un internaute

Un internaute peut s'inscrire sur l'application puis se connecter grâce à un identifiant et un mot de passe. Il peut ré-initialiser son mot de passe qui lui serait envoyé par mail.

Connexion

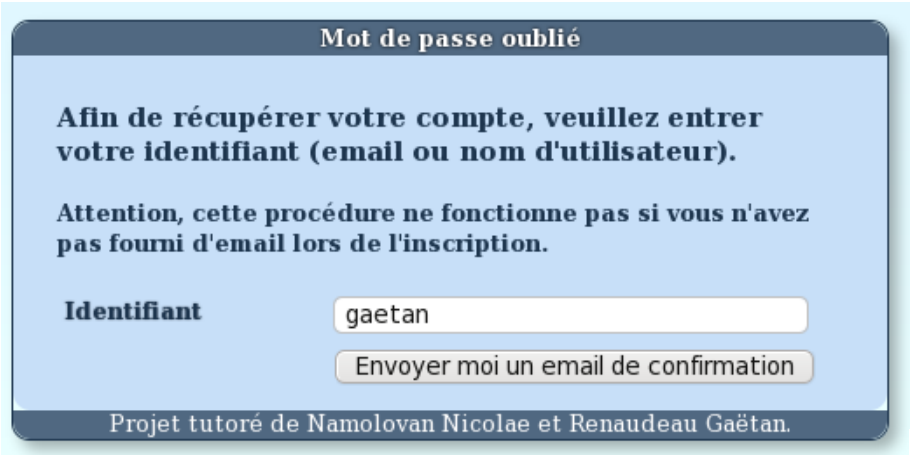
Inscription

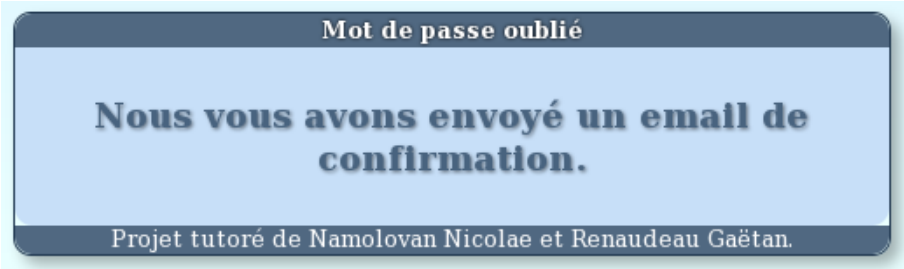


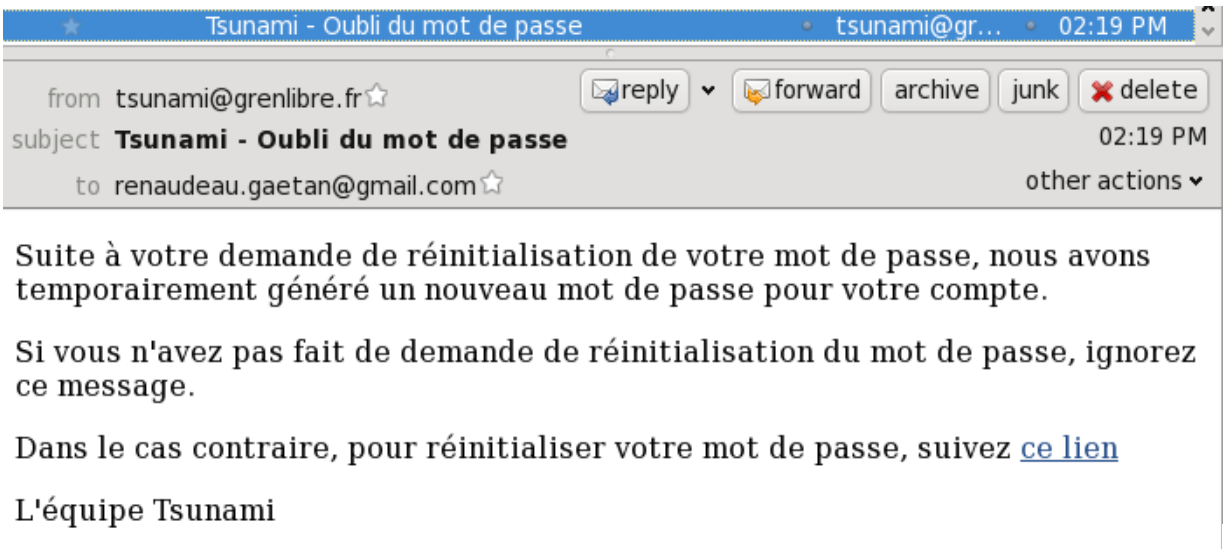
Récupération du mot de passe par mail

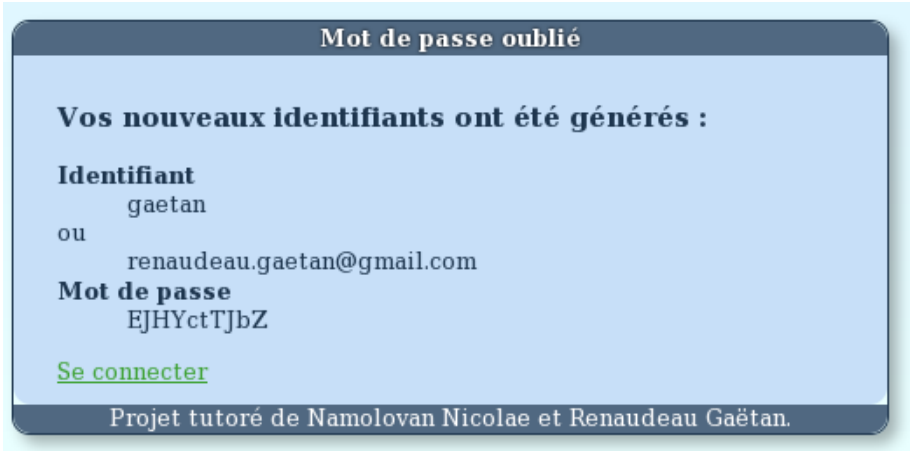
Un utilisateur peut récupérer son mot de passe s'il l'a oublié.

Procédure de récupération du mot de passe









Gestion du profil et des préférences

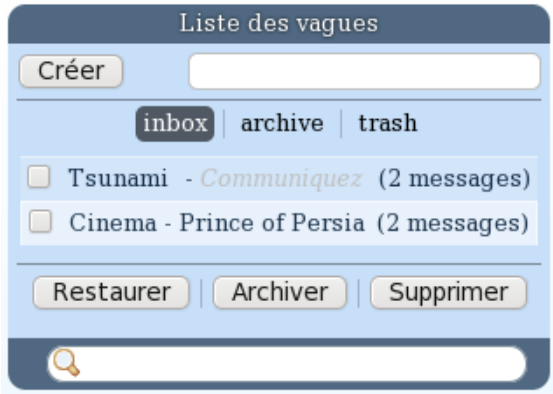
Un utilisateur peut modifier son message personnel, un message qui sera visible par tous ses amis, son status (disponible, occupé, absent, ...) et son avatar.
Il peut également modifier son mot de passe ou encore personaliser son interface (la couleur du thème, redimension des fenêtres).

Gestion des contacts

Chaque utilisateur possède une liste de contacts. L'ajout de nouveaux utilisateurs à entrer dans sa liste de contact est géré par un système d'invitation. Les contacts peuvent être organisés en groupes. Le status d'un contact indique si un contact est disponible, occupé, absent, inactif ou déconnecté. On peut facilement déplacer un contact d'un groupe à l'autre au moyen d'un glissé-déposé.

Gestion des vagues

Les discussions, appelées vagues, sont gérés dans une liste. Les vagues contenant des nouveaux messages sont affichés en bold. Il est possible de créer une vague, de l'afficher et l'éditer, de la mettre à la corbeille. Il est également possible d'effectuer une recherche dans sa liste de vagues. La mise à jour de la recherche se fait de façon dynamique.



Gestion des messages d'une vague

Une discussion est dynamique :

- La succession des messages d'une discussion n'est pas linéaire : Il est possible de créer des sous discussions au sein des messages d'une discussion. On peut ainsi la caractériser comme un arbre de messages.
- Un utilisateur est libre d'éditer tous les messages d'une discussion même les messages ne lui appartenant pas.
- Il est également possible de voir l'historique d'une discussion, c'est à dire l'évolution des messages au cours du temps.
- Deux utilisateurs peuvent éditer le même message simultanément, les mises à jour sont réalisées instantanément.

Administration

Une administration permet aux administrateurs d'accéder aux tuples de la table User.

Exigences non fonctionnelles

L'ergonomie

Pour une bonne expérience utilisateur, il est nécessaire que l'application respecte les critères d'ergonomie suivants :

L'homogénéité

- Le thème de l'application permet une homogénéité des fenêtres, des liens, des titres ...
- Les formulaires sont identiques (même style pour les champs, les messages d'erreurs, ...).

La concision

Réduction des activités de perception, de compréhension, de mémorisation et de formulation d'une suite d'actions pour réaliser une intention.

La signifiance

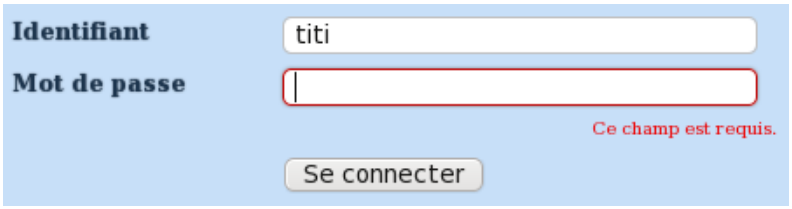
Capacité d'un élément de l'IHM à être significatif pour les utilisateurs.

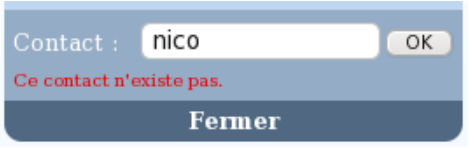
La barre de status sur l'application indique le chargement des composants de Tsunami.

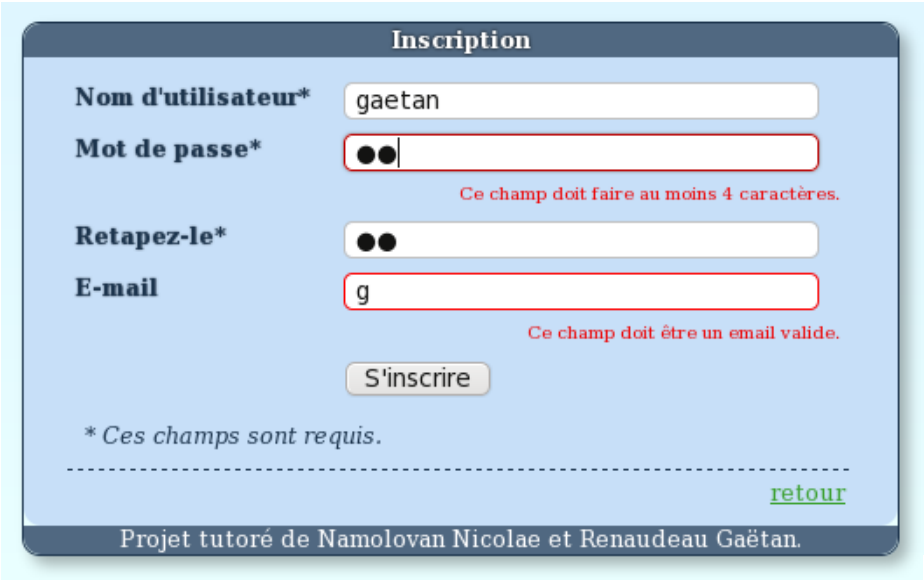
La rétroaction

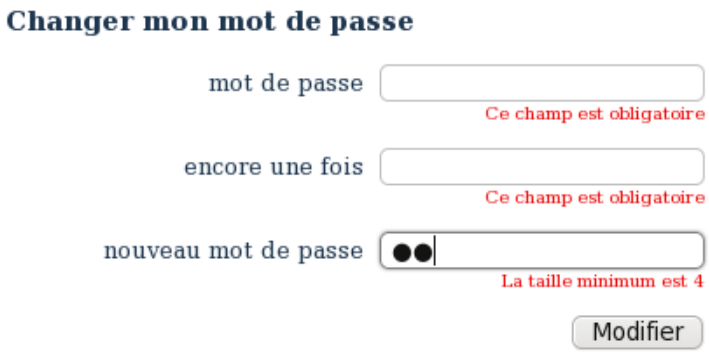
Démonstration de la prise en compte des actions effectuées.

Un message d'erreur précis apparait sous chaque champ pour informer des erreurs de validation d'un formulaire.









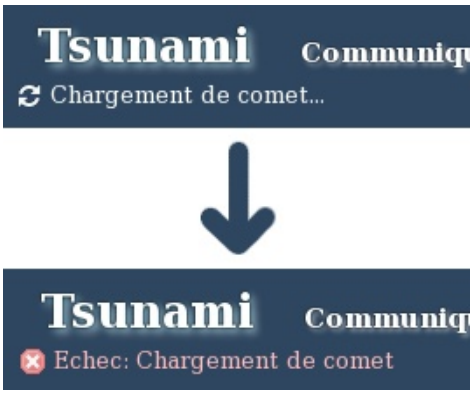
La flexibilité

Aptitude à prendre en compte les aléas et aptitude à prendre en compte l'expérience des utilisateurs.

- La possibilité de redimensionner la largeur des fenêtres et de choisir un thème permettent d'améliorer l'expérience des utilisateurs

- Si le javascript n'est pas activé sur le navigateur de l'utilisateur, ou que notre application n'est pas compatible avec ce navigateur, un message alternatif s'affiche sur la page d'accueil.

Lorsqu'un problème technique rend l'application incapable de fonctionner, un message apparait au chargement :



La compatibilité

L'application a pour objectif d'être compatible avec tous les navigateurs récents (HTML 5) et respectant les standards du W3C.

La sécurité

- le cryptage des mots de passe
- la validation des arguments passés aux contrôleurs et la vérification des droits (utilisation des codes d'erreur du HTTP tels que 403 Forbidden, 404 Not Found, ...)

Présentation technique

Diagramme de classe

Ceci est le diagramme de classe récupéré après reverse-engineering du schéma de la base (Java) et après quelques améliorations.

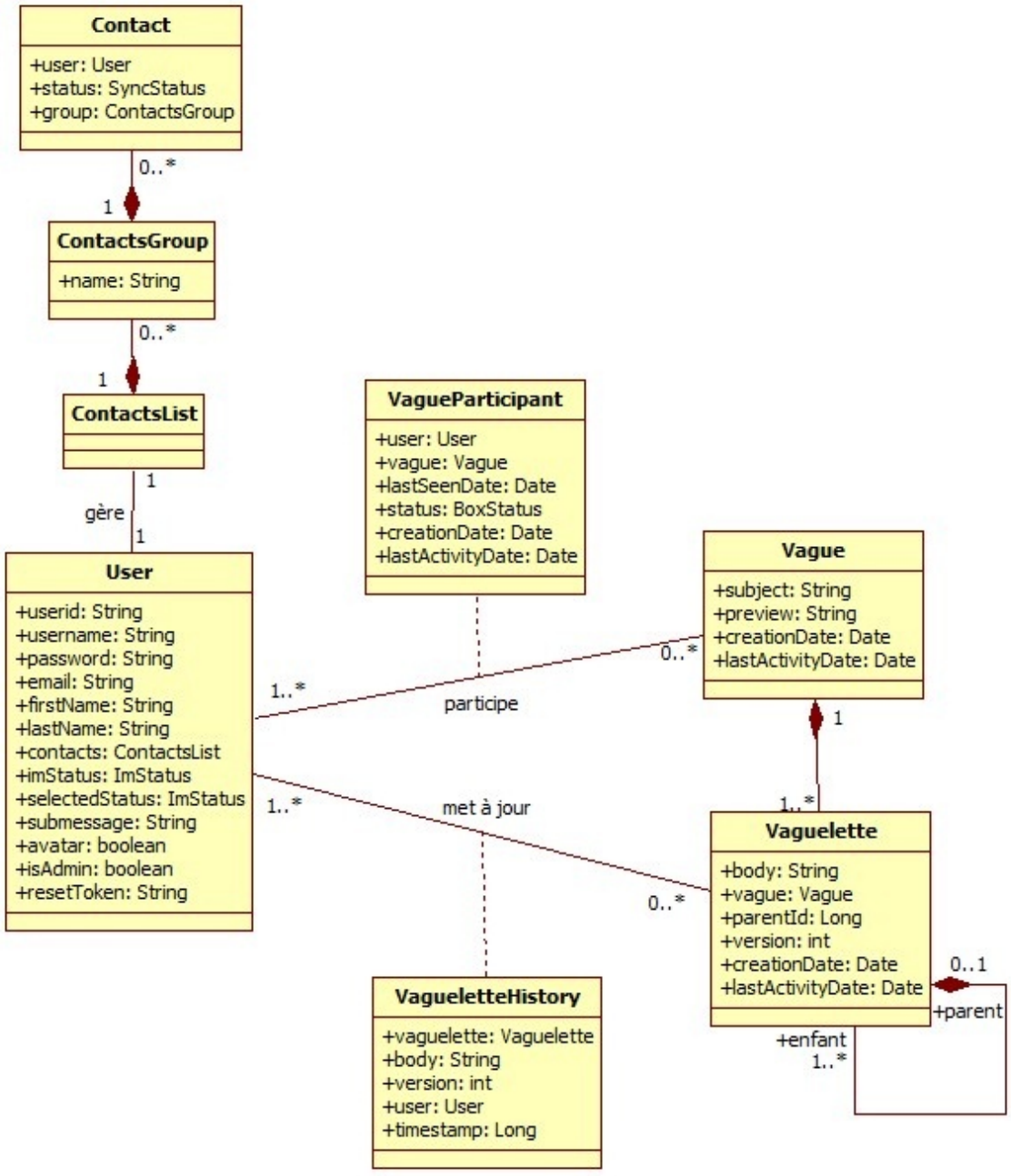
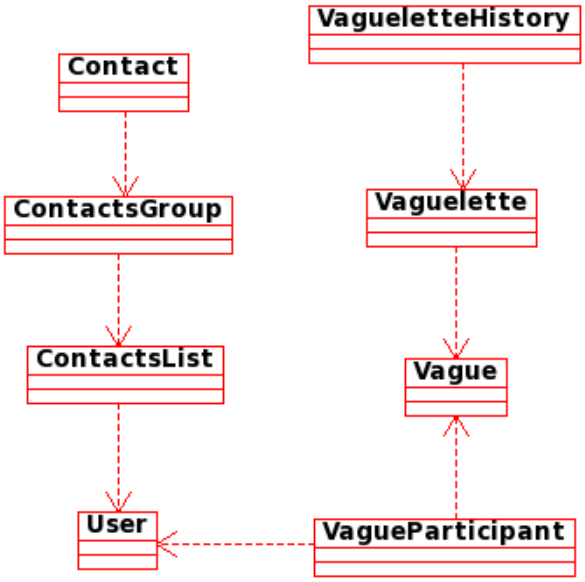
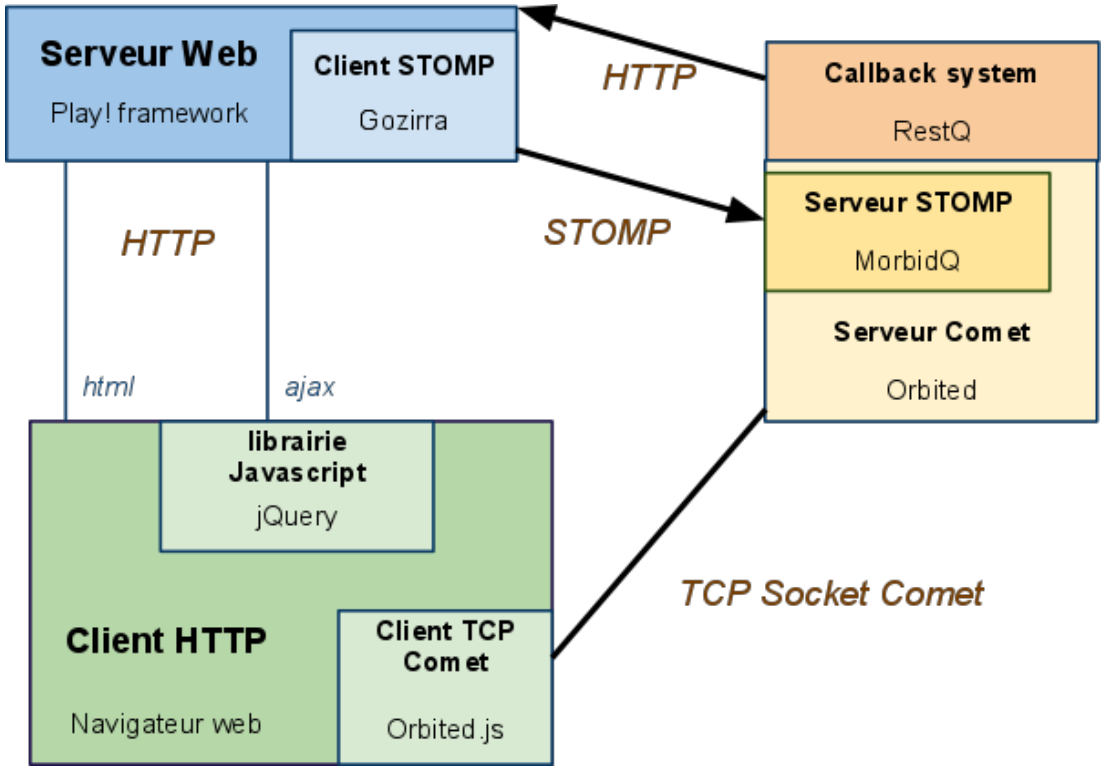


Schéma simplifié de la base



Architecture technique de Tsunami

Le web à ses débuts ne prévoyait pas de réaliser des applications aussi dynamiques comme on le voit depuis quelques années. La technologie s'est beaucoup améliorer avec le Javascript permettant notamment d'effectuer des requêtes Ajax et le HTML 5 permettant entre autre de faire du socket avec la technologie WebSocket. Néanmoins, cette technologie étant nouvelle et en perpétuelle évolution (draft W3C) et de plus pas supporté par tous les navigateurs, nous devions utiliser des bibliothèques assurant une compatibilité et une stabilité raisonnable.



Serveur web

Nous avons choisi le framework web **Play! framework** écrit en Java.
Dans notre application, Play! framework nous permet d'utiliser le modèle MVC : Model-View-Controller :

- **Model** : gérer les données (possibilité de connecter une base de donnée).
- **View** : réaliser l'interface de l'application en HTML et JSON.
- **Controller** : associer une URI à une action.

Serveur STOMP

C'est un middleware orienté "messages". Il fournit un format interopérable qui permet aux clients STOMP de dialoguer avec n'importe quel fournisseur de message supportant le protocole.

Callback system

Tous les événements qui se passe sur le serveur STOMP, sont communiqué au framework web grâce au "callback system".
Dans notre cas, nous utilisons ce système pour traiter les évènements connexion et déconnexion des utilisateurs afin de mettre à jour le champ "statut" de client dans la base de données.
Techniquement, on crée une requête http vers le serveur web en passant l'id de l'utilisateur ainsi que son nouveau statut.

Serveur comet

Le serveur comet permet d'envoyer des informations au navigateur web sans que celui-ci l'ait explicitement demandé.
Le navigateur peut alors recevoir les informations dès qu'ils sont disponibles.

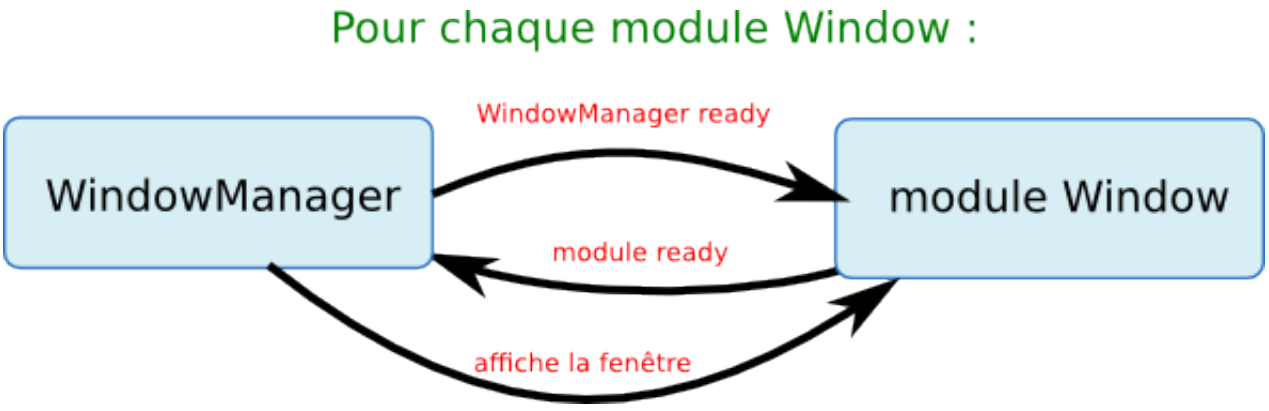
Client Comet

Chaque client web est abonnés ("subscribe") à une flux associé à un utilisateur. La communication passe par le protocole STOMP.

Au chargement de l'application, le client comet crée une connexion socket avec le serveur comet puis déclanche le chargement du **gestionnaire de fenêtres**.

Fonctionnement des fenêtres

L'application est organisé en fenêtres (Window) qui sont gérées par le gestionnaire de fenêtres (WindowManager). Notre application comporte 3 fenêtres : la liste des contacts, la liste des vagues, l'affichage d'une vague.
Au chargement de l'application, le gestionnaires de fenêtres, une fois chargé, déclanche le chargement des fenêtres. Une fois que toutes les fenêtres se sont chargés, le gestionnaire de fenêtres les affiche.



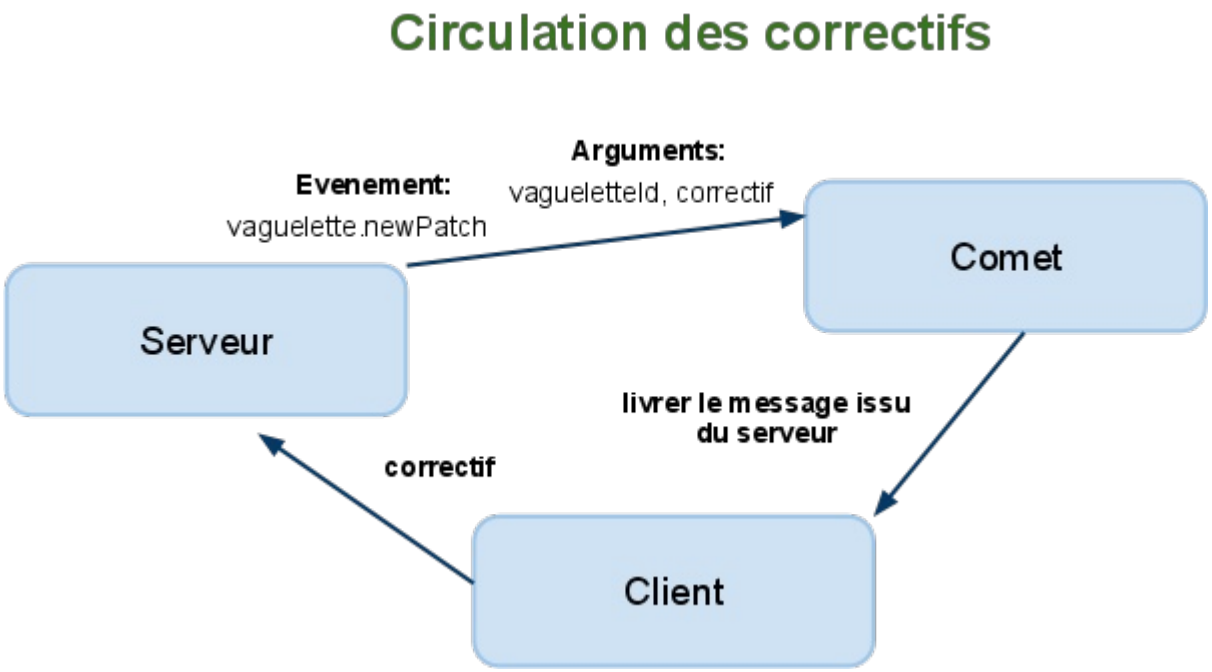
C'est également le gestionnaire de fenêtres qui gère les poignées entre les fenêtres.

Fonctionnement de la synchronisation

Synchronisation du texte d'une vaguelette

La synchronisation d'un texte est fait avec des **correctifs** (patches).

- **côté client**, au changement du texte par l'utilisateur, des correctifs sont générés puis envoyé au serveur.
- **Le serveur** applique le correctif à sa version de vaguelette et **diffuse** le même correctif à toute les parties concernées.
- Les autres clients appliquent les correctifs reçus.



Comment les clients sont-ils synchronisés ?

Par la connexion Comet, il reçoit des données dans un format spécifique, constituant du nom de l'événement, et les paramètres nécessaires pour ce type d'événement.
Par exemple, quand un nouveau participant est ajouté à une vague, l'événement *vague.newParticipant* avec les arguments *vagueId* et *userId* est envoyé à tout les autres participants de la même vague.

Fonctionnement de l'historisation d'une vaguelette

Tous les correctifs reçus sont enregistrés les uns à la suite des autres afin de concerver toute la séquence des changements depuis la création de la vaguelette.
Grâce à ces enregistrements, l'application peut construire l'historique des changements d'une vaguelette pas à pas afin de pouvoir revenir en arrière.

Intérêt du projet et montée en compétences

Le travail collaboratif

- Google documents a été utilisé pour établir le cahier des charges et ce présent rapport ainsi que pour s'organiser (liste de tâches) durant tout le projet.
- L'itilisation d'un gestionnaire de sources (svn puis git utilisés) a permis un meilleur suivi des évolutions du code source.
- Utilisation de google docs pour partager une liste de tâches à effectuer (todo list).

Compétences techniques

La synchronisation

Compétences web

HTML 5

Nous avons expérimenté l'utilisation du HTML 5 encore en cours d'élaboration pour notre projet (nouvelles balises, animation canvas, ...).

Javascript (modulaire)

Le javascript modulaire consiste à organiser ses scripts en modules indépendants. Nous utilisons les évènements javascript afin de faire communiquer les modules entre eux.
La bibliothèque jQuery permet de gérer facilement les évènements javascript.

CSS 3 et SASS

Nous avons expérimenté l'utilisation du CSS 3 qui apporte de formidables améliorations pour styliser un site comme les dégradés, les ombres, les bordures arrondis, ...

De plus nous avons utilisé le Sass (Syntactically Awesome Stylesheets), un langage amélioré du css qui permet de factoriser beaucoup de code css et de rendre son écriture et sa maintenance rapide et moins contraignante. Il est compilé en css (compilé à la volée grâce à un module pour le framework play).

Son utilisation a de nombreux avantages par rapport au CSS :

- la simplicité (plus de crochets, plus de point virgule mais juste de l'indentation)
- l'imbrication des sélecteurs css (appliquant l'idée du *DRY : don't repeat yourself*)
- l'utilisation de variables
- l'utilisation d'opérations élémentaires (sur les pixels, les couleurs, ...)
- la factorisation de code (au lieu de faire des copier-coller, on peux factoriser le code à travers les "mixins" avec possibilité de passer des arguments).
- La réduction du code css et la possibilité de tout mettre dans un fichier (via l'héritage)

Ce langage n'est pas difficile à apprendre, cela ressemble au css, avec de nombreuses fonctionnalités intéressantes en plus.

Le framework play!

Nous avons choisi le framework web **Play! framework** écrit en Java. C'est un framework qui permet facilement d'utiliser REST (Representational State Transfer).

Le langage Java

Nous avons donc appris à manier le langage Java en même temps qu'on l'étudiait en cours.

Le JPA (Java Persistence API)

Le JPA est une interface de programmation Java permettant d'organiser des données relationnelles. C'est une barrière d'abstraction qui évite l'utilisation du SQL et permet une simplicité d'écriture. De plus, cela permet de manipuler directement les tuples de la base de données en tant qu'objet, de pouvoir les sauvegarder, les supprimer...
Exemple :
en JPA : *User.findById(1)*
en SQL : *select * from User where id = 1;*

Conclusion

En s'inspirant du tout nouveau Google Wave, nous avons décidé de reprendre et d'approfondir certaines fonctionnalités et d'en oublier certaines qui aurait été long à mettre en oeuvre, je pense en particulier à la dimension Rich Text d'une discussion sur Google Wave.
Nous nous sommes focalisé sur la gestion du temps réel, les performances, et avons tenter de respecter aux mieux les standards du web afin de montrer la nouvelle dimension des applications web, aujourd'hui, en 2010.

Lexique

Reverse Engineering ou Rétro-ingénierie

La rétro-ingénierie est l'activité qui consiste à étudier un objet pour en déterminer le fonctionnement interne ou la méthode de fabrication.

REST

REST (**Representational State Transfer**) est une manière de construire une application pour les systèmes distribués comme le World Wide Web.
REST est un style d'architecture dans laquelle, un composant lit ou modifie une ressource en utilisant une représentation de cette ressource. Une ressource est une chose nommable, qui peut évoluer avec le temps. Une représentation est une séquence d'octets, éventuellement accompagnée de métadonnées (séquence de nom/valeur). Un composant est un acteur, il est relié à d'autres composants et à des ressources par des canaux, qui permettent des interactions sans états.

MVC

Le **Modèle-Vue-Contrôleur** (en abrégé **MVC**, de l'anglais *Model-View-Controller*) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle. Ce paradigme divise l'IHM en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation), chacun ayant un rôle précis dans l'interface.

Middleware

Un **middleware** (anglicisme) est une couche de logiciels qui crée un réseau d'échange d'informations entre différentes applications informatiques.

URI

Un **URI**, de l'anglais ***Uniform Resource Identifier***, soit littéralement *identifiant uniforme de ressource*, est une courte chaîne de caractères identifiant une ressource sur un réseau (par exemple une ressource Web) physique ou abstraite, et dont la syntaxe respecte une norme d'Internet mise en place pour le World Wide Web.

Framework

En programmation informatique, un **framework** est un *kit* de composants logiciels structurels, qui définissent les fondations ainsi que les grandes lignes de l'organisation de tout ou partie d'un logiciel (architecture). En programmation orientée objet un framework est typiquement composé de classes mères qui seront dérivées et étendues par héritage en fonction des besoins spécifiques à chaque logiciel qui utilise le framework.

Références

- http://wikipedia.fr/
- http://wave.google.com/
- http://playframework.org/

Déploiement

Instructions techniques expliquant le déploiement de l'application.

Composants requis

- Système (**linux*** ou mac) disposant d'une machine virtuelle **java** http://java.sun.com/ et d'un interpréteur **python**.
- Le framework Play! http://www.playframework.org/
- Un navigateur correct (compatibilité assurée sous **Firefox*** (3.6 conseillé) et Chrome) http://www.mozilla-europe.org/fr/firefox/

* Les pré-requis recommandés sont indiqués en gras.

Lancement de l'application

Nous avons automatisé le lancement de l'application dans un script bash.
Depuis les sources de l'application. Lancez le grâce à la commande **`./run`**
Il se charge de :

- Télécharger play s'il n'est pas trouvé dans le PATH
- Lancer le serveur HTTP (play) qui se chargera de lancer le serveur comet (restq et orbited).

Vous n'aurez alors plus qu'à vous rendre sur la page indiquée (dans la configuration par défaut sur http://localhost:8000/). Au premier chargement, et uniquement à ce moment, play va compiler l'application.